

METHODS FOR DETERMINING THE SIMILARITY  
OF CONTENT AND STRUCTURING UNSTRUCTURED  
CONTENT FROM HETEROGENEOUS SOURCES

List of Inventors

1. Dr. David S. Warren, Ph.D  
112 Christian Avenue, Stony Brook, NY 11790  
United States Citizen
2. Dr. Terrance L. Swift, Ph.D  
826 Leigh Mill Road, Great Falls, VA 22066  
United States Citizen
3. Tatyana Vidrevich  
30 Reeves Road, Port Jefferson, NY 11777  
United States Citizen
4. Dr. IV Ramakrishnan, Ph.D.  
4 High Gate Drive, Setauket, NY  
United States Citizen
5. L. Robert Pokorny  
2267 River Road, Calverton, NY 11933  
United States Citizen
6. Alex Beggs  
6 University Drive, Setauket, NY 11733  
United States Citizen
7. Christopher Rued  
22 Campsite Lane, East Setauket, NY 11733  
United States Citizen
8. Michael Epstein  
14 Alma Lane, East Northport, NY 11731  
United States Citizen
9. Harpreet Singh  
234 Doherty Avenue, Elmont, NY 11003  
Citizen of India
10. Dr. Hasan Davulcu, Ph.D  
1064 San Palmilla Apartments, 750 W. Baseline Road, Tempe, AZ 85283-1107  
United States Citizen

1                   IN THE UNITED STATES  
2                   PATENT AND TRADEMARK OFFICE  
3                   PATENT APPLICATION  
4

5                   METHODS FOR DETERMINING THE SIMILARITY  
6                   OF CONTENT AND STRUCTURING UNSTRUCTURED  
7                   CONTENT FROM HETEROGENEOUS SOURCES  
8

9           The Federal Government shall have a non-exclusive,  
10 nontransferable, irrevocable, paid-up license to practice or  
11 have practiced for or on behalf of the United States the subject  
12 invention throughout the world as provided for by SBIR Grant No.  
13 0128508 awarded by the National Science Foundation.  
14

15           This application claims the benefit of provisional  
16 application serial number 60/410,684, filed September 13, 2002,  
17 the complete disclosure of which is hereby incorporated by  
18 reference.  
19

20                   BACKGROUND OF THE INVENTION  
21

22   1. Field of the Invention

23           The invention relates to researching and organizing  
24 information from a plurality of sources. More particularly, the

invention relates to computer assisted mining and organization of information from electronic sources.

## 2. Brief Description of the Prior Art

Never in the history of humanity has there been so much information available to so many people. The advent of the World Wide Web in the early 1990s created the ability to access information stored in computer databases all over the world from any computer connected to the PSTN (public switched telephone network). According to the Online Computer Library Center, Inc. (<http://wcp.oclc.org/>), there were approximately 2,851,000 web sites in 1998 and approximately 8,712,000 in 2002. Although growth has slowed, the number of websites is still increasing every year.

Many websites contain little or no useful information. However, there are also many websites which contain a wealth of valuable information. The difficulty is in locating and organizing the available information. Many so-called search engines attempt to organize the content of the World Wide Web. The most well known are, perhaps, Yahoo and Google. While these search engines are helpful for the casual user, they are incomplete and often inaccurate. Moreover, information retrievable from the Internet is not formatted in a standard

uniform structure. For example, data may be in HTML format, PDF format, Microsoft Word (.doc) format, tab-delimited format, XML format, etc. Even information found in the same document format are often presented in various sources. For example, data may be tabled in some sources, and described in free text in others. Additionally, different lexicons are often used to describe the same features. Thus, in order to mine information for use in a queriable database, the information must be restructured to a uniform view.

Businesses have always recognized that accurate, precise, coherent data is a powerful tool for making sound business decisions. Many businesses have realized that extremely valuable information can be mined from the World Wide Web as well as from other Internet resources such as "news groups" and "ftp sites" and from their own electronic data. However, successfully retrieving and organizing this information is costly and time consuming. The state of the art approach is to employ skilled data and domain experts to manually extract, classify, structure and categorize data. This process can take up to an hour for a single data entry. In addition to information mined from the Internet, it would be desirable to integrate that information with existing "legacy data" in a company's own electronic file system. Much of this data is only

semi-structured, e.g. tabular data in a text document, or  
completely unstructured, e.g. free flowing text.

#### SUMMARY OF THE INVENTION

It is therefore an object of the invention to facilitate  
the retrieval and organization of information from multiple  
locations and types of data sources.

It is also an object of the invention to facilitate the  
retrieval and organization of information from multiple data  
sources connected to the Internet.

It is still another object of the invention to facilitate  
the retrieval and organization of information from multiple data  
sources connected to the World Wide Web.

It is yet another object of the invention to facilitate the  
restructuring of information to a uniform format for use in a  
queriable database.

It is also an object of the invention to facilitate the  
structuring of unstructured and semi-structured information to a  
uniform format for use in a queriable database.

1       It is still another object of the invention to provide  
2 methods of matching and determining the similarity of data.

3  
4       Accordingly, the methods of the present invention include  
5 acquiring data of interest, creating a knowledge base which  
6 represents the semantics of the domain of interest, using the  
7 knowledge base to categorize the content of the acquired data  
8 and to infer values of its attributes, and comparing and  
9 quantifying the similarity of the data content.

10  
11       The step of acquiring (the acquisition process) is example  
12 driven. That is, the user provides examples of the data to be  
13 extracted (mined), preferably via an easy to use graphical  
14 interface. An algorithm is provided which uses the examples to  
15 access and extract data from specified sources, to infer meaning  
16 from the extracted data, and to rapidly structure the data into  
17 a useful format.

18  
19       The invention provides several software tools for acquiring  
20 and organizing information. The tools include a web agent  
21 creator for creating web agents or "bots" which penetrate  
22 websites and harvest desired information. These bots are  
23 capable of following links and filling in forms to reach desired  
24 information buried deep in a website. A text extractor tool is

1 also provided to extract tabular data from text documents. The  
2 text extractor uses a clustering algorithm to determine column  
3 and row delimiters of tables embedded in text documents. In  
4 order to classify information, the invention provides an  
5 ontology management system and an ontology directed classifier.  
6 The ontology management system stores and manages classes and  
7 their relationships as well as objects and their attributes.  
8 The ontology directed classifier has four stages: taxonomy  
9 token weighting, node weighting for descriptors, weight  
10 propagation and normalization, and determining the best class  
11 and cone, a sub-tree of the taxonomy. An ontology directed  
12 extractor enables automated extraction of attribute-value pairs  
13 from a textual description of an object based on ontology  
14 knowledge of a class to which the object was classified.  
15 Lastly, the invention provides a validation component. The  
16 validation component utilizes ANSI/ASQC Z1.4-1993 "Sampling  
17 Procedures and Tables for Inspection by Attributes", which is  
18 applied iteratively on random samples. An acceptable quality  
19 level (AQL) is selected and compared to a random sample. If the  
20 sample has fewer defects than the AQL, a lower AQL is selected  
21 and applied to another random sample. The process is repeated  
22 until a sample with more defects than the selected AQL is found.  
23 The last successful AQL is taken to be accurate. If the first  
24 random sample has more defects than the selected AQL, the AQL is

1 raised and applied to another random sample. This is repeated  
2 until a sample is found to pass the AQL and that AQL is taken to  
3 be accurate.

4  
5 The tools of the invention operate independently and  
6 together, e.g. the output of one tool providing input for  
7 another tool.

8  
9 BRIEF DESCRIPTION OF THE DRAWINGS

10  
11 FIG. 1 is a schematic diagram of a web agent creator  
12 according to the invention;

13  
14 FIG. 2 is a schematic illustration illustrating the  
15 concepts of the XPath discovery algorithm of the web agent  
16 creator;

17  
18 FIG. 3 is a screen shot of the main user interface of the  
19 web agent creator;

20  
21 FIGS. 4a-4d are screen shots illustrating the operation of  
22 the web agent creator;



FIG. 5 is an illustration of semi-structured tabular data having no column delimiters;

FIGS. 6a and 6b are screen shots illustrating the operation of a text extractor tool;

FIG. 7 is a screen shot illustrating the data of FIG. 5 after structuring by the text extractor tool;

FIG. 8 is a fragment of a taxonomy as used in the ontology management system;

FIG. 9 is a screen shot illustrating the operation of the ontology directed classifier;

FIGS. 10 and 11 are screen shots illustrating the construction of an ontology directed extractor;

FIG. 12 is a screen shot illustrating data acquired by the ontology data extractor; and

FIG. 13 is a simplified flow chart illustrating how all of the tools of the invention function together.

## BRIEF DESCRIPTION OF THE APPENDICES

Appendix A is a document (21 pages including Table of Contents) describing an ontology-directed matcher according to the invention.

Appendix B is a document (23 pages including Table of Contents) comprising a user guide to ontology-directed matcher software according to the invention.

The attached CDROM appendix includes source code for the tools of the invention. The CDROM is in ISO 9660 format and contains the following files:

Name	Size	Last Modified
XSB_SOURCE090903	36,803,028	
:AgentManager	2,965,893	
::com	2,965,893	
:::xsb	2,965,893	
::::LibConstants.java	332	5/15/02 11:21 AM
:::::extraction	55,909	
:::::tableextractor	55,909	
:::::AncestorList.java	1,150	11/26/01 7:50 AM
:::::CheckValidTable.java	2,053	11/26/01 7:49 AM
:::::ConvertTablesToList.java	3,276	11/26/01 7:48 AM
:::::CreateTables.java	19,217	11/26/01 2:53 PM
:::::DOMTableExtractor.java	10,639	11/28/01 12:26 PM
:::::DataActionObject.java	879	11/26/01 7:40 AM
:::::DataObject.java	1,021	11/26/01 11:04 AM
:::::ExtractTableFromDomTree.java	3,550	11/26/01 7:49 AM
:::::GenerateKeywordList.java	1,150	11/27/01 5:37 PM
:::::TableExtractor.java	1,476	11/28/01 12:27 PM
:::::TableExtractorException.java	617	11/29/01 9:07 AM
:::::TagNames.java	481	11/26/01 7:48 AM
:::::TraverseDOMTree.java	10,400	11/28/01 12:26 PM
:::::images	14,019	
:::::ani_robot.gif	974	1/21/02 9:10 PM
:::::bandage16x16.gif	860	5/1/02 12:11 PM

1	:::::help.gif	161	6/25/03 12:11 PM
2	:::::map16x16.gif	935	9/28/01 4:03 PM
3	:::::mbclosed.gif	1,308	5/1/02 12:11 PM
4	:::::mbopen.gif	1,299	5/1/02 12:11 PM
5	:::::microscope16x16.gif	877	5/1/02 12:11 PM
6	:::::right_arrow.gif	822	7/30/02 9:46 AM
7	:::::xrover_ode.jpg	5,892	5/1/02 12:11 PM
8	:::::xsb_icon_16_16_trans.gif	891	5/1/02 12:11 PM
9	:::::launcher	623,039	
10	:::::AddDialog.form	7,778	1/4/01 6:07 PM
11	:::::AddDialog.java	12,216	7/24/03 10:08 AM
12	:::::AddFileDialog.form	9,652	1/4/01 6:07 PM
13	:::::AddFileDialog.java	8,884	1/24/01 7:21 AM
14	:::::AdvancedPrefPanel.form	14,117	6/30/03 1:41 PM
15	:::::AdvancedPrefPanel.java	17,674	7/1/03 10:17 AM
16	:::::ArtificialGlobalsInputSourceList.java	854	7/24/03 10:08 AM
17	:::::Constants.java	12,045	7/24/03 10:08 AM
18	:::::Constants_prob.java	11,699	7/24/03 10:07 AM
19	:::::EmailConfigPanel.form	5,759	10/9/01 8:03 AM
20	:::::EmailConfigPanel.java	9,122	7/24/03 10:08 AM
21	:::::Launcher.form	11,303	9/12/02 4:25 PM
22	:::::Launcher.java	53,298	7/24/03 10:08 AM
23	:::::LogWPanel.java	18,650	7/24/03 10:08 AM
24	:::::NewLauncher.form	7,741	8/20/03 1:48 PM
25	:::::NewLauncher.java	55,579	8/20/03 1:48 PM
26	:::::OutputWPanel.form	346	1/4/01 6:07 PM
27	:::::OutputWPanel.java	46,264	7/24/03 10:08 AM
28	:::::PDFFileWPanel.java	31,590	7/24/03 10:08 AM
29	:::::PongPanel.form	266	10/30/01 6:40 AM
30	:::::PongPanel.java	7,121	10/30/01 6:40 AM
31	:::::QueryWPanel.java	16,760	7/24/03 10:08 AM
32	:::::ScheduleEditorPanel.form	5,807	7/18/03 11:29 AM
33	:::::ScheduleEditorPanel.java	17,238	7/24/03 10:08 AM
34	:::::SitePlanWPanel.java	7,838	7/24/03 10:08 AM
35	:::::SitemapCatalogPrefPanel.form	4,600	6/30/03 1:33 PM
36	:::::SitemapCatalogPrefPanel.java	8,127	7/1/03 10:37 AM
37	:::::TaskCatalogPanel.form	2,047	1/29/03 2:13 PM
38	:::::TaskCatalogPanel.java	6,747	10/14/02 12:53 PM
39	:::::TaskDirectoryPrefPanel.form	4,949	6/26/03 10:49 AM
40	:::::TaskDirectoryPrefPanel.java	10,023	7/1/03 10:23 AM
41	:::::TaskInputPanel.form	12,440	7/18/03 1:32 PM
42	:::::TaskInputPanel.java	45,647	7/24/03 10:08 AM
43	:::::TaskLogPanel.form	1,132	10/9/01 7:50 AM
44	:::::TaskLogPanel.java	1,566	10/9/01 7:50 AM
45	:::::TaskOutputPanel.form	29,397	6/20/03 1:07 PM
46	:::::TaskOutputPanel.java	39,938	6/20/03 1:07 PM
47	:::::TaskSchedulerPanel.form	1,126	10/30/01 6:40 AM
48	:::::TaskSchedulerPanel.java	3,225	10/30/01 6:40 AM
49	:::::TaskTreeTableModel.java	8,002	11/15/01 4:07 PM
50	:::::TaskTreeTableNode.java	5,860	7/17/03 6:03 AM
51	:::::TaskWizard.form	6,161	4/26/02 2:17 PM
52	:::::TaskWizard.java	13,667	7/11/02 10:24 AM
53	:::::TaskWizardDialog.form	1,897	10/30/01 6:39 AM

1	::::TaskWizardDialog.java	5,089	7/24/03 10:08 AM
2	::::TaskWizardPanel.form	520	9/12/02 1:26 PM
3	::::TaskWizardPanel.java	6,050	1/30/03 8:34 AM
4	::::XRouterTaskJTreeTable.java	5,089	7/24/03 10:08 AM
5	::::images	10,139	
6	:::::icon.gif	1,375	12/10/02 5:41 AM
7	:::::splash.jpg	7,873	6/3/03 9:35 AM
8	:::::xsb_icon.gif	891	1/10/01 8:52 AM
9	::::ui	376,528	
10	::::AboutDialog.form	2,176	7/10/03 11:37 AM
11	::::AboutDialog.java	4,165	7/10/03 11:37 AM
12	::::AboutPanel.form	2,673	7/10/03 11:34 AM
13	::::AboutPanel.java	2,892	7/10/03 11:34 AM
14	::::AlternateColorJTable.java	1,806	7/24/03 9:58 AM
15	::::ApplicationInformation.java	17,914	7/10/03 6:43 AM
16	::::AttributableTableModel.java	9,505	5/27/03 12:04 PM
17	::::AuthProxyPreferencesPanel.form	12,175	6/30/03 1:46 PM
18	::::AuthProxyPreferencesPanel.java	30,639	7/24/03 9:58 AM
19	::::CachingJComboBox.java	3,823	7/24/03 9:58 AM
20	::::CertificatePolicyPanel.form	2,011	9/13/01 11:44 AM
21	::::CertificatePolicyPanel.java	2,064	7/10/03 6:45 AM
22	::::DirectorySelectorPanel.form	2,258	6/20/03 11:36 AM
23	::::DirectorySelectorPanel.java	5,926	6/20/03 11:36 AM
24	::::FileViewTree.java	6,768	7/24/03 9:58 AM
25	::::FileViewTreeField.java	8,798	7/24/03 12:31 PM
26	::::FileViewTreeModel.java	6,597	7/24/03 9:58 AM
27	::::HostnameField.java	1,201	7/24/03 9:58 AM
28	::::ImagePanel.form	529	6/2/03 9:30 AM
29	::::ImagePanel.java	3,091	6/3/03 9:45 AM
30	::::LookAndFeelJMenu.java	4,574	7/24/03 9:58 AM
31	::::NonEditableTableModel.java	936	7/24/03 9:58 AM
32	::::PasswordComboBoxEditor.java	3,166	6/23/03 5:59 AM
33	::::PreferencesEditor.java	17,876	7/24/03 9:58 AM
34	::::PreferencesPanel.java	2,484	7/24/03 9:58 AM
35	::::SchedulerPanel.form	8,102	2/15/02 7:14 AM
36	::::SchedulerPanel.java	17,844	7/24/03 9:58 AM
37	::::SizzleButton.java	2,730	11/6/01 10:34 AM
38	::::SplashPanel.form	6,761	6/3/03 9:45 AM
39	::::SplashPanel.java	8,313	6/9/03 1:36 PM
40	::::SplashScreen.java	5,402	6/9/03 1:36 PM
41	::::TimeChooserDialog.form	2,885	6/23/03 6:48 AM
42	::::TimeChooserDialog.java	5,264	7/24/03 9:58 AM
43	::::TimeChooserPanel.form	9,654	7/8/01 5:15 PM
44	::::TimeChooserPanel.java	16,768	7/24/03 9:58 AM
45	::::UIUtils.java	21,907	7/24/03 9:58 AM
46	::::WizardTabView.form	974	4/19/01 9:30 PM
47	::::WizardTabView.java	3,528	7/24/03 12:31 PM
48	::::XSBDirectorySelectorDialog.form	4,215	11/9/01 2:19 PM
49	::::XSBDirectorySelectorDialog.java	5,889	11/9/01 2:19 PM
50	::::XSBWizardPanel.java	2,727	7/24/03 9:58 AM
51	::::webbrowser	97,518	
52	:::::MozillaParserPromptSupport.java	2,345	5/14/02 8:04 AM
53	:::::MozillaUtils.java	19,636	6/5/03 8:17 AM

1	:::::dom	75,537		
2	:::::AbstractMagnet.java	10,470	7/24/03	9:58 AM
3	:::::AnchorMagnet.java	3,593	7/24/03	9:58 AM
4	:::::DefaultAnchorMagnet.java	7,099	7/24/03	9:58 AM
5	:::::DefaultTextMagnet.java	1,838	7/24/03	9:58 AM
6	:::::ERExpression.java	9,606	7/24/03	9:58 AM
7	:::::Isolator.java	3,976	7/24/03	9:58 AM
8	:::::Magnet.java	4,793	7/24/03	9:58 AM
9	:::::MagnetFactory.java	1,963	7/24/03	9:58 AM
10	:::::MozillaParserImpl.java	19,508	7/31/03	1:04 PM
11	:::::NodeListImpl.java	3,116	7/24/03	9:58 AM
12	:::::Parser.java	734	7/24/03	9:58 AM
13	:::::TextMagnet.java	801	7/24/03	9:58 AM
14	:::::xpath	8,040		
15	:::::XPathException.java	1,135	7/24/03	9:58 AM
16	:::::XPathProcessor.java	3,575	7/24/03	9:58 AM
17	:::::XPathProcessorImplOmQuery.java	3,330	7/24/03	9:58 AM
18	:::::util	361,323		
19	:::::AbstractConsumerProcessor.java	1,117	7/24/03	12:31 PM
20	:::::AbstractMessenger.java	2,774	7/24/03	9:58 AM
21	:::::ClassInfo.java	937	7/24/03	9:58 AM
22	:::::CommandLineParser.java	3,533	7/24/03	9:58 AM
23	:::::Constant.java	2,299	7/24/03	9:58 AM
24	:::::ConsumerPool.java	22,286	7/24/03	9:58 AM
25	:::::ConsumerProcessor.java	2,180	7/24/03	9:58 AM
26	:::::DBUtils.java	36,537	7/24/03	9:58 AM
27	:::::DataStructureUtils.java	9,500	7/24/03	9:58 AM
28	:::::DefaultFileFilter.java	4,752	7/24/03	9:58 AM
29	:::::DefaultMessenger.java	3,513	7/24/03	9:58 AM
30	:::::DevUtils.java	4,222	7/24/03	9:58 AM
31	:::::HTTPContext.java	1,023	10/2/02	7:20 AM
32	:::::IOUtils.java	32,332	9/3/03	10:25 AM
33	:::::JSUtils.java	1,698	3/5/03	11:38 AM
34	:::::License.java	1,224	7/24/03	9:58 AM
35	:::::LogEntry.java	2,092	7/24/03	9:58 AM
36	:::::MapEntryImpl.java	1,232	7/17/03	6:08 AM
37	:::::MemoryQueue.java	2,829	7/24/03	9:58 AM
38	:::::MessageConstant.java	1,182	7/24/03	9:58 AM
39	:::::Messenger.java	5,836	7/24/03	9:58 AM
40	:::::ObservableImpl.java	443	7/16/03	10:42 AM
41	:::::Queue.java	3,495	7/24/03	9:58 AM
42	:::::RegularExpression.java	3,791	7/24/03	9:58 AM
43	:::::SharedLock.java	12,196	1/8/02	8:22 AM
44	:::::SoftHashMap.java	20,896	7/24/03	9:58 AM
45	:::::StringUtils.java	32,122	8/20/03	1:59 PM
46	:::::TemporalLicense.java	3,364	7/24/03	9:58 AM
47	:::::TrimConstant.java	1,731	7/24/03	9:58 AM
48	:::::Utils.java	44,739	7/24/03	9:58 AM
49	:::::WebUtils.java	15,310	6/26/03	1:17 PM
50	:::::html	5,960		
51	:::::AuthenticatingProxyConfig.java	3,856	5/14/02	8:01 AM
52	:::::ProxyConfig.java	2,104	7/24/03	9:58 AM
53	:::::regexp	27,456		

1	Expression.java	4,707	7/24/03 9:58 AM
2	Match.java	2,809	7/24/03 9:58 AM
3	RegExprException.java	1,877	7/24/03 9:58 AM
4	RegularExpression.java	16,841	7/24/03 9:58 AM
5	SubExpressionMatch.java	1,222	7/24/03 9:58 AM
6	xmlserialization	46,722	
7	XMLDeserializationException.java	1,296	7/24/03 9:58 AM
8	XMLSerializable.java	2,836	7/24/03 9:58 AM
9	XMLSerializationException.java	1,253	7/24/03 9:58 AM
10	XMLSerializer.java	38,089	7/24/03 9:58 AM
11	XMLSerializerException.java	1,266	7/24/03 9:58 AM
12	XMLSerializerObjectCache.java	1,982	7/24/03 9:58 AM
13	xml	57,493	
14	dom	57,493	
15	DOMUtils.java	50,191	9/5/03 7:50 AM
16	adapters	7,302	
17	KeyValuePairToAttrNodeAdapter.java	4,152	6/9/03 1:55 PM
18	MapToNamedNodeMapAdapter.java	3,150	6/9/03 1:51 PM
19	xml2dbms	136,181	
20	MapConstants.java	933	8/22/01 11:31 AM
21	Xml2Dbms.java	40,328	7/24/03 9:58 AM
22	Xml2DbmsException.java	554	1/14/02 9:49 AM
23	imagefiles	158	
24	srcfile.gif	79	11/30/98 7:55 PM
25	textfile.gif	79	11/30/98 7:56 PM
26	initializationfiles	94,208	
27	blankdb.mdb	94,208	8/9/01 7:24 AM
28	xrover	1,341,069	
29	AbstractAction.java	17,686	7/31/03 1:42 PM
30	AbstractDataContainer.java	58,793	7/24/03 9:58 AM
31	AbstractFilter.java	3,733	7/24/03 9:58 AM
32	AbstractFilterGroup.java	2,855	7/24/03 9:58 AM
33	Action.java	6,310	6/19/03 7:21 AM
34	ActionConstants.java	740	7/24/03 9:57 AM
35	ActionFactory.java	1,683	10/4/02 6:32 AM
36	Argument.java	2,195	7/24/03 9:58 AM
37	DataContainer.java	3,666	6/19/03 7:55 AM
38	DataContainerFactory.java	8,198	7/14/03 7:42 AM
39	DataDefinitionCollection.java	2,324	7/24/03 9:57 AM
40	DataDefinitionCollectionFactory.java	3,463	12/19/02 2:06 PM
41	DataDefinitionConstants.java	2,027	7/24/03 9:56 AM
42	DataObjectFileNotFoundException.java	1,137	7/24/03 9:58 AM
43	DataTypes.java	4,195	7/24/03 12:31 PM
44	DatabaseDataContainerImpl.java	75,513	11/4/02 12:17 PM
45	DatabaseDataDefinitionCollection.java	6,464	11/11/02 5:56 AM
46	DatabaseRootDataContainerImpl.java	9,741	10/10/02 10:19 AM
47	DefaultActionImpl.java	6,836	7/24/03 9:58 AM
48	DefaultDataContainerImpl.java	9,293	7/8/03 9:57 AM
49	DefaultFilterGroup.java	2,919	7/24/03 9:57 AM
50	DefaultGlobal.java	2,450	1/30/03 2:52 PM
51	DefaultNumberFilter.java	6,825	7/24/03 9:58 AM
52	DefaultRetriever.java	18,607	9/9/03 1:07 PM
53	DefaultRootDataContainerImpl.java	8,832	7/24/03 9:58 AM

1	.....DefaultStringFilter.java	8,738	7/24/03 9:58 AM
2	.....DtdFileNotFoundException.java	1,109	7/24/03 9:58 AM
3	.....EvaluationException.java	649	2/24/03 12:39 PM
4	.....ExtractionErrorException.java	1,056	7/24/03 9:58 AM
5	.....Filter.java	1,111	7/24/03 9:58 AM
6	.....FilterGroup.java	1,577	7/24/03 9:58 AM
7	.....Global.java	1,089	1/30/03 8:06 AM
8	.....GlobalConstants.java	3,482	8/12/03 7:11 AM
9	.....InvalidMapException.java	753	11/29/01 9:02 AM
10	.....MalformedPlanException.java	1,828	7/24/03 9:58 AM
11	.....NoSignatureMatchException.java	1,060	7/24/03 9:58 AM
12	.....NumberFilterOperatorConstant.java	1,407	7/24/03 9:57 AM
13	.....PageID.java	3,455	7/24/03 9:58 AM
14	.....PageMapInterpreter.java	147,594	8/18/03 10:27 AM
15	.....ProcessingException.java	1,022	7/24/03 9:58 AM
16	.....Retriever.java	5,605	9/9/03 1:07 PM
17	.....RootDataContainer.java	694	9/4/02 3:38 PM
18	.....SiteMapInterpreter.java	84,477	8/5/03 7:45 AM
19	.....SiteMapInterpreterException.java	1,804	7/24/03 9:58 AM
20	.....SitePathTagConstants.java	6,654	7/24/03 9:58 AM
21	.....StringFilterOperatorConstant.java	2,296	7/24/03 9:58 AM
22	.....UndefinedDataTypeException.java	1,064	7/24/03 9:58 AM
23	.....cluster	119,468	
24	.....scheduler	119,468	
25	.....DatabaseGlobalsInputSource.java	13,349	7/24/03 9:58 AM
26	.....DefaultSingleGlobalsInputSource.java	2,417	7/11/03
27	5:56 AM		
28	.....GlobalsInputSource.java	812	7/24/03 12:31 PM
29	.....GlobalsInputSourceException.java	1,137	7/24/03 9:58 AM
30	.....GlobalsInputSourceList.java	3,434	11/2/01 2:48 PM
31	.....ManagerImpl.java	40,836	7/17/03 7:17 AM
32	.....MapGlobalsInputSource.java	3,477	7/24/03 9:58 AM
33	.....MultipleGlobalsInputSource.java	794	7/24/03 9:58 AM
34	.....Scheduler.java	13,374	7/24/03 9:58 AM
35	.....SchedulingQueue.java	6,408	7/24/03 9:58 AM
36	.....SingleGlobalsInputSource.java	1,195	7/24/03 9:58 AM
37	.....WaitingQueue.java	6,032	7/24/03 9:58 AM
38	.....XRoverTask.java	15,197	7/24/03 9:58 AM
39	.....XRoverTaskConsumer.java	2,077	7/24/03 9:58 AM
40	.....XRoverTaskConsumerException.java	709	7/24/03 9:58 AM
41	.....XRoverTaskManager.java	4,050	6/5/03 12:01 PM
42	.....XRoverTaskManagerEvent.java	552	1/14/02 8:09 AM
43	.....XRoverTaskManagerListener.java	1,247	1/14/02 8:09 AM
44	.....XRoverTaskObserver.java	258	10/19/01 6:43 AM
45	.....XRoverTaskProducer.java	1,404	7/24/03 9:58 AM
46	.....XRoverTaskProducerException.java	709	7/24/03 9:58 AM
47	.....plugins	40,989	
48	.....ERExpressionPlugin.java	27,220	7/24/03 9:58 AM
49	.....HTMLTableExtractorPlugin.java	12,834	6/19/03 12:43 PM
50	.....Plugin.java	935	7/24/03 9:58 AM
51	.....util	635,603	
52	.....AbstractDataContainerExporter.java	9,629	7/24/03 9:58 AM
53	.....DOMParser.java	40,324	7/24/03 9:58 AM

1	::::::DataContainer.mdb	331,776	2/3/03 10:49 AM
2	::::::DataContainer2BarSeparated.java	5,549	7/24/03 9:58 AM
3	::::::DataContainer2CSV.java	3,849	7/24/03 9:58 AM
4	::::::DataContainer2HTML.java	6,013	7/24/03 9:58 AM
5	::::::DataContainer2TableModel.java	19,279	7/24/03 9:58 AM
6	::::::DataContainer2XML.java	2,624	7/24/03 9:58 AM
7	::::::DataContainerCompletedObserver.java	1,740	7/24/03 9:58 AM
8	::::::DataContainerDocumentAdapter.java	19,195	10/4/02 7:46 AM
9	::::::DataContainerExport.java	1,728	7/24/03 9:58 AM
10	::::::DataContainerExporter.java	3,079	7/24/03 9:58 AM
11	::::::DataContainerLeafNodeAdapter.java	4,216	6/9/03 1:36 PM
12	::::::DataContainerList.java	2,216	6/9/03 2:04 PM
13	::::::DataContainerNodeAdapter.java	42,940	6/2/03 11:11 AM
14	::::::DataContainerTextAdapter.java	10,240	5/7/03 6:30 AM
15	::::::DataContainerUtils.java	10,983	6/9/03 2:08 PM
16	::::::DataDefinitionCollectionCache.java	3,846	7/24/03 9:58 AM
17	::::::DatabaseDataContainerExporter.java	6,504	7/24/03 9:58 AM
18	::::::DefaultDataContainerExport.java	3,326	7/24/03 9:58 AM
19	::::::ExportException.java	876	7/24/03 9:58 AM
20	::::::FilterTreeModel.java	4,686	7/24/03 9:58 AM
21	::::::InputReader.java	17,405	7/24/03 9:58 AM
22	::::::Notifier.java	602	8/22/02 7:29 AM
23	::::::RootDataContainerObserver.java	3,127	7/24/03 9:58 AM
24	::::::SessionInfoBuilder.java	3,739	5/13/03 9:37 AM
25	::::::SessionInfoBuilderZipImpl.java	7,542	6/2/03 12:03 PM
26	::::::SessionInfoBuilderZipMergedImpl.java	3,364	6/2/03 11:50 AM
27	::::::SessionInfoConstants.java	682	5/5/03 10:01 AM
28	::::::SessionInfoZipImplConstants.java	2,210	5/5/03 8:07 AM
29	::::::TableModel2CSV.java	7,042	7/24/03 9:58 AM
30	::::::TableModel2CharSeparated.java	10,855	7/24/03 9:58 AM
31	::::::XML2DataDefinitionCollection.java	13,018	7/24/03 9:58 AM
32	::::::XMLDataContainerExporter.java	7,979	7/24/03 9:58 AM
33	::::::XMLException.java	1,077	7/24/03 9:58 AM
34	::::::XRoverDatabaseFactory.java	5,697	1/16/03 7:58 AM
35	::::::XRoverUtilities.java	15,558	7/24/03 9:58 AM
36	::::::XSLException.java	1,088	7/24/03 9:58 AM
37	:EasyRover	1,707,442	
38	::com	1,707,442	
39	:::xsb	1,707,442	
40	::::LibConstants.java	332	5/15/02 11:21 AM
41	::::easyrover	165,063	
42	:::::AttributeEditor.form	6,641	1/18/02 1:50 AM
43	:::::AttributeEditor.java	15,854	2/20/02 6:43 AM
44	:::::Constants.java	8,964	6/6/03 1:22 PM
45	:::::ERProxyPreferencesPanel.java	2,131	6/7/02 1:10 PM
46	:::::EasyRover.java	73,987	6/23/03 10:14 AM
47	:::::FormEditorFrame.java	1,799	2/27/02 6:55 AM
48	:::::LimitedMozillaPromptSupport.java	3,456	6/10/02 7:23 AM
49	:::::PathExtractionEditorDialog.java	16,156	6/23/03 10:12 AM
50	:::::PathNameDocument.java	1,316	10/24/02 12:58 PM
51	:::::RobotsDialog.form	5,810	2/20/02 7:57 AM
52	:::::RobotsDialog.java	6,409	2/20/02 7:57 AM
53	:::::images	22,540	



1	anim_tree.gif	3,134	1/17/02 9:44 PM
2	flashingRedLED.gif	181	8/19/02 6:21 AM
3	flashing_caution.gif	144	1/17/02 9:44 PM
4	icon.gif	1,395	12/10/02 5:41 AM
5	kiwi	747	
6	blank.gif	100	1/17/02 9:44 PM
7	caution.gif	138	1/17/02 9:44 PM
8	led-green-on.gif	142	1/17/02 9:44 PM
9	led-off.gif	140	1/17/02 9:44 PM
10	led-red-on.gif	142	1/17/02 9:44 PM
11	no.small.gif	85	1/17/02 9:44 PM
12	led-yellow-on.gif	890	1/17/02 9:44 PM
13	red_caution.gif	885	1/18/02 3:05 AM
14	robots_warning.gif	399	1/17/02 9:44 PM
15	splash.jpg	14,631	6/6/03 1:22 PM
16	tree.gif	134	1/17/02 9:44 PM
17	extraction	55,909	
18	tableextractor	55,909	
19	AncestorList.java	1,150	11/26/01 7:50 AM
20	CheckValidTable.java	2,053	11/26/01 7:49 AM
21	ConvertTablesToList.java	3,276	11/26/01 7:48 AM
22	CreateTables.java	19,217	11/26/01 2:53 PM
23	DOMTableExtractor.java	10,639	11/28/01 12:26 PM
24	DataActionObject.java	879	11/26/01 7:40 AM
25	DataObject.java	1,021	11/26/01 11:04 AM
26	ExtractTableFromDomTree.java	3,550	11/26/01 7:49 AM
27	GenerateKeywordList.java	1,150	11/27/01 5:37 PM
28	TableExtractor.java	1,476	11/28/01 12:27 PM
29	TableExtractorException.java	617	11/29/01 9:07 AM
30	TagNames.java	481	11/26/01 7:48 AM
31	TraverseDOMTree.java	10,400	11/28/01 12:26 PM
32	images	14,019	
33	ani_robot.gif	974	1/21/02 9:10 PM
34	bandage16x16.gif	860	5/1/02 12:11 PM
35	help.gif	161	6/25/03 12:11 PM
36	map16x16.gif	935	9/28/01 4:03 PM
37	mbclosed.gif	1,308	5/1/02 12:11 PM
38	mbopen.gif	1,299	5/1/02 12:11 PM
39	microscope16x16.gif	877	5/1/02 12:11 PM
40	right_arrow.gif	822	7/30/02 9:46 AM
41	xrover_ode.jpg	5,892	5/1/02 12:11 PM
42	xsb_icon_16_16_trans.gif	891	5/1/02 12:11 PM
43	ui	348,680	
44	AttributableTableModel.java	9,505	5/27/03 12:04 PM
45	DirectorySelectorPanel.form	2,258	6/20/03 11:36 AM
46	DirectorySelectorPanel.java	5,926	6/20/03 11:36 AM
47	FileViewTree.java	6,768	7/24/03 9:58 AM
48	FileViewTreeModel.java	6,597	7/24/03 9:58 AM
49	HostnameField.java	1,201	7/24/03 9:58 AM
50	ImagePanel.form	529	6/2/03 9:30 AM
51	ImagePanel.java	3,091	6/3/03 9:45 AM
52	PreferencesEditor.java	17,876	7/24/03 9:58 AM
53	PreferencesPanel.java	2,484	7/24/03 9:58 AM

1	:::::ProxyPreferencesPanel.form	4,175	6/6/02 6:45 AM
2	:::::ProxyPreferencesPanel.java	10,064	7/24/03 9:58 AM
3	:::::SplashPanel.form	6,761	6/3/03 9:45 AM
4	:::::SplashPanel.java	8,313	6/9/03 1:36 PM
5	:::::SplashScreen.java	5,402	6/9/03 1:36 PM
6	:::::UIUtils.java	21,907	7/24/03 9:58 AM
7	:::::WizardTabView.form	974	4/19/01 9:30 PM
8	:::::WizardTabView.java	3,528	7/24/03 12:31 PM
9	:::::XSBDirectorySelectorDialog.form	4,215	11/9/01 2:19 PM
10	:::::XSBDirectorySelectorDialog.java	5,889	11/9/01 2:19 PM
11	:::::XSBWizardPanel.java	2,727	7/24/03 9:58 AM
12	:::::webbrowser	218,490	
13	:::::MozillaBrowserFrame.java	35,353	9/9/03 1:11 PM
14	:::::MozillaParserPromptSupport.java	2,345	5/14/02 8:04 AM
15	:::::MozillaPromptSupport.java	3,696	5/14/02 8:03 AM
16	:::::MozillaUtils.java	19,636	6/5/03 8:17 AM
17	:::::RenderedView.java	782	7/24/03 9:58 AM
18	:::::dom	156,678	
19	:::::AbstractMagnet.java	10,470	7/24/03 9:58 AM
20	:::::AnchorMagnet.java	3,593	7/24/03 9:58 AM
21	:::::DOMSelectionHandler.java	15,288	8/28/03 2:34 PM
22	:::::DefaultAnchorMagnet.java	7,099	7/24/03 9:58 AM
23	:::::DefaultTextMagnet.java	1,838	7/24/03 9:58 AM
24	:::::ERConstants.java	3,950	7/24/03 9:58 AM
25	:::::ERExpression.java	9,606	7/24/03 9:58 AM
26	:::::EasyRover.java	9,461	7/24/03 9:58 AM
27	:::::Isolator.java	3,976	7/24/03 9:58 AM
28	:::::Magnet.java	4,793	7/24/03 9:58 AM
29	:::::MagnetFactory.java	1,963	7/24/03 9:58 AM
30	:::::MapSerializableArgument.java	1,136	11/5/01 2:06 PM
31	:::::MozillaParserImpl.java	19,508	7/31/03 1:04 PM
32	:::::NodeHighlightSupport.java	8,355	9/9/03 1:07 PM
33	:::::NodeListImpl.java	3,116	7/24/03 9:58 AM
34	:::::Parser.java	734	7/24/03 9:58 AM
35	:::::TextMagnet.java	801	7/24/03 9:58 AM
36	:::::XPathExpression.java	42,951	7/24/03 9:58 AM
37	:::::xpath	8,040	
38	:::::XPathException.java	1,135	7/24/03 9:58 AM
39	:::::XPathProcessor.java	3,575	7/24/03 9:58 AM
40	:::::XPathProcessorImplOmQuery.java	3,330	7/24/03 9:58 AM
41	:::::util	259,514	
42	:::::AbstractMessenger.java	2,774	7/24/03 9:58 AM
43	:::::ClassInfo.java	937	7/24/03 9:58 AM
44	:::::CommandLineParser.java	3,533	7/24/03 9:58 AM
45	:::::Constant.java	2,299	7/24/03 9:58 AM
46	:::::DBUtils.java	36,537	7/24/03 9:58 AM
47	:::::DataStructureUtils.java	9,500	7/24/03 9:58 AM
48	:::::DefaultMessenger.java	3,513	7/24/03 9:58 AM
49	:::::DevUtils.java	4,222	7/24/03 9:58 AM
50	:::::HTTPContext.java	1,023	10/2/02 7:20 AM
51	:::::IOUtils.java	32,332	9/3/03 10:25 AM
52	:::::JSUtils.java	1,698	3/5/03 11:38 AM
53	:::::License.java	1,224	7/24/03 9:58 AM

1	:::::MessageConstant.java	1,182	7/24/03 9:58 AM
2	:::::Messenger.java	5,836	7/24/03 9:58 AM
3	:::::Queue.java	3,495	7/24/03 9:58 AM
4	:::::RegularExpression.java	3,791	7/24/03 9:58 AM
5	:::::SoftHashMap.java	20,896	7/24/03 9:58 AM
6	:::::StringUtils.java	32,122	8/20/03 1:59 PM
7	:::::TemporalLicense.java	3,364	7/24/03 9:58 AM
8	:::::TrimConstant.java	1,731	7/24/03 9:58 AM
9	:::::Utils.java	44,739	7/24/03 9:58 AM
10	:::::WebUtils.java	15,310	6/26/03 1:17 PM
11	:::::regexp	27,456	
12	:::::Expression.java	4,707	7/24/03 9:58 AM
13	:::::Match.java	2,809	7/24/03 9:58 AM
14	:::::RegExprException.java	1,877	7/24/03 9:58 AM
15	:::::RegularExpression.java	16,841	7/24/03 9:58 AM
16	:::::SubExpressionMatch.java	1,222	7/24/03 9:58 AM
17	:::::xml	125,853	
18	:::::dom	125,853	
19	:::::DOMUtils.java	50,191	9/5/03 7:50 AM
20	:::::FormDataPanel.form	9,214	8/8/02 7:12 AM
21	:::::FormDataPanel.java	18,519	7/24/03 9:58 AM
22	:::::FormInfo.java	40,627	6/27/02 5:33 AM
23	:::::adapters	7,302	
24	:::::KeyValuePairToAttrNodeAdapter.java	4,152	6/9/03 1:55 PM
25	:::::MapToNamedNodeMapAdapter.java	3,150	6/9/03 1:51 PM
26	:::::xml2dbms	136,181	
27	:::::MapConstants.java	933	8/22/01 11:31 AM
28	:::::Xml2Dbms.java	40,328	7/24/03 9:58 AM
29	:::::Xml2DbmsException.java	554	1/14/02 9:49 AM
30	:::::imagefiles	158	
31	:::::srcfile.gif	79	11/30/98 7:55 PM
32	:::::textfile.gif	79	11/30/98 7:56 PM
33	:::::initializationfiles	94,208	
34	:::::blankdb.mdb	94,208	8/9/01 7:24 AM
35	:::::xrover	601,891	
36	:::::AbstractAction.java	17,686	7/31/03 1:42 PM
37	:::::AbstractDataContainer.java	58,793	7/24/03 9:58 AM
38	:::::Action.java	6,310	6/19/03 7:21 AM
39	:::::ActionConstants.java	740	7/24/03 9:57 AM
40	:::::ActionFactory.java	1,683	10/4/02 6:32 AM
41	:::::Argument.java	2,195	7/24/03 9:58 AM
42	:::::DataContainer.java	3,666	6/19/03 7:55 AM
43	:::::DataContainerFactory.java	8,198	7/14/03 7:42 AM
44	:::::DataDefinitionCollection.java	2,324	7/24/03 9:57 AM
45	:::::DataDefinitionConstants.java	2,027	7/24/03 9:56 AM
46	:::::DataTypes.java	4,195	7/24/03 12:31 PM
47	:::::DatabaseDataContainerImpl.java	75,513	11/4/02 12:17 PM
48	:::::DatabaseDataDefinitionCollection.java	6,464	11/11/02 5:56 AM
49	:::::DatabaseRootDataContainerImpl.java	9,741	10/10/02 10:19 AM
50	:::::DefaultActionImpl.java	6,836	7/24/03 9:58 AM
51	:::::DefaultDataContainerImpl.java	9,293	7/8/03 9:57 AM
52	:::::DefaultRetriever.java	18,607	9/9/03 1:07 PM
53	:::::DefaultRootDataContainerImpl.java	8,832	7/24/03 9:58 AM

1	:::::DtdFileNotFoundException.java	1,109	7/24/03 9:58 AM
2	:::::EvaluationException.java	649	2/24/03 12:39 PM
3	:::::ExtractionErrorException.java	1,056	7/24/03 9:58 AM
4	:::::GlobalConstants.java	3,482	8/12/03 7:11 AM
5	:::::InvalidMapException.java	753	11/29/01 9:02 AM
6	:::::MalformedPlanException.java	1,828	7/24/03 9:58 AM
7	:::::PageID.java	3,455	7/24/03 9:58 AM
8	:::::PageMapInterpreter.java	147,594	8/18/03 10:27 AM
9	:::::ProcessingException.java	1,022	7/24/03 9:58 AM
10	:::::Retriever.java	5,605	9/9/03 1:07 PM
11	:::::RootDataContainer.java	694	9/4/02 3:38 PM
12	:::::SitePathTagConstants.java	6,654	7/24/03 9:58 AM
13	:::::SourcedArgument.java	1,215	7/19/02 5:54 AM
14	:::::UndefinedDataTypeException.java	1,064	7/24/03 9:58 AM
15	:::::plugins	40,989	
16	:::::ERExpressionPlugin.java	27,220	7/24/03 9:58 AM
17	:::::HTMLTableExtractorPlugin.java	12,834	6/19/03 12:43 PM
18	:::::Plugin.java	935	7/24/03 9:58 AM
19	:::::util	141,619	
20	:::::DOMParser.java	40,324	7/24/03 9:58 AM
21	:::::DataContainerDocumentAdapter.java	19,195	10/4/02 7:46 AM
22	:::::DataContainerLeafNodeAdapter.java	4,216	6/9/03 1:36 PM
23	:::::DataContainerNodeAdapter.java	42,940	6/2/03 11:11 AM
24	:::::DataContainerTextAdapter.java	10,240	5/7/03 6:30 AM
25	:::::Notifier.java	602	8/22/02 7:29 AM
26	:::::SessionInfoConstants.java	682	5/5/03 10:01 AM
27	:::::XMLException.java	1,077	7/24/03 9:58 AM
28	:::::XRouterDatabaseFactory.java	5,697	1/16/03 7:58 AM
29	:::::XRouterUtilities.java	15,558	7/24/03 9:58 AM
30	:::::XSLEException.java	1,088	7/24/03 9:58 AM
31	:ode_oms	1,108,707	
32	:::attribute_parser.P	803	1/22/03 9:59 AM
33	:::config_ode_template.P	3,985	5/13/03 6:54 AM
34	:::main.P	2,622	4/7/02 6:54 AM
35	:::ode_classifier	197,174	
36	:::images	1,314	
37	:::icon_classifier.gif	1,314	11/5/02 7:30 AM
38	:::ode_classifierGUIConcept.P	33,048	8/4/03 4:42 AM
39	:::ode_classifierGUIExplain.P	10,897	4/23/03 6:37 AM
40	:::ode_classifierGUIObjList.P	21,340	3/28/03 5:46 AM
41	:::ode_classifierGUISearch.P	17,208	8/4/03 4:42 AM
42	:::ode_classifierGUITraining.P	12,506	3/18/03 10:12 AM
43	:::ode_classifierGUIUtils.P	12,088	5/13/03 7:32 AM
44	:::ode_classifierGUIdatactr.P	897	12/12/02 6:32 AM
45	:::ode_classifierGUIinval_dyn.P	4,325	5/19/03 5:49 AM
46	:::ode_classifierGUImain.P	57,562	8/4/03 4:42 AM
47	:::ode_classifierGUIt2t.P	16,667	5/13/03 7:07 AM
48	:::ode_classifierValidation.P	9,142	5/19/03 9:54 AM
49	:::preferences_classifier.P	180	9/9/03 11:57 AM
50	:::ode_defaults.P	1,092	6/13/03 12:25 PM
51	:::ode_domain_types.P	9,842	6/13/03 11:28 AM
52	:::ode_domain_types_cdf.P	10,356	5/19/03 10:39 AM
53	:::ode_dtl.P	3,304	4/7/02 6:54 AM

1	::ode_editor	76,169	
2	:::ode_editorGUIAttribute.P	9,475	6/18/03 9:32 AM
3	:::ode_editorGUIAttributeObject.P	9,212	6/18/03 9:32 AM
4	:::ode_editorGUIConcept.P	24,898	2/24/03 1:03 PM
5	:::ode_editorGUIEditor.P	6,550	5/8/03 1:11 PM
6	:::ode_editorGUIObject.P	10,218	2/19/03 4:48 AM
7	:::ode_editorGUIRelationship.P	11,820	1/24/03 10:55 AM
8	:::ode_editorGUIedit.P	3,904	4/15/03 4:54 AM
9	:::preferences_editor.P	92	6/25/03 10:57 AM
10	::ode_formatoms.P	12,246	6/12/03 7:43 AM
11	::ode_genrepsoms.P	23,103	6/18/03 6:39 AM
12	::ode_init.P	2,223	4/15/03 10:05 AM
13	::ode_initcdf.P	2,917	5/14/03 1:32 PM
14	::ode_launcher	125,383	
15	:::attributelist.P	7,064	9/9/03 11:33 AM
16	:::attributelistcdf.P	7,195	5/19/03 2:00 PM
17	:::clstree.P	13,992	5/20/03 11:04 AM
18	:::clstreecdf.P	14,615	5/20/03 11:36 AM
19	:::domain_gen.P	100	1/20/03 9:49 AM
20	:::extractor.P	14,974	4/24/03 7:31 AM
21	:::images	22,308	
22	:::about.html	353	6/5/03 11:47 AM
23	:::icon_odelauncher.gif	1,307	11/5/02 7:30 AM
24	:::icon_odelauncher.jpg	1,420	10/31/02 11:32 AM
25	:::odel_splash.jpg	19,228	6/5/03 11:38 AM
26	:::import_batch.P	9,160	3/28/03 1:18 PM
27	:::ode_launcher_io.P	17,044	4/21/03 6:59 AM
28	:::ode_validator.P	2,762	9/9/03 11:31 AM
29	:::replacements_gen.P	1,323	1/20/03 9:49 AM
30	:::settings.P	1,803	9/20/02 6:00 AM
31	:::source_oms_info.P	2,771	5/22/03 6:45 AM
32	:::tableview.P	5,349	4/9/03 6:09 AM
33	:::xjode.P	4,923	5/9/03 7:26 AM
34	::ode_nclassifier.P	31,565	5/13/03 7:07 AM
35	::ode_parser.P	14,477	6/11/03 7:03 AM
36	::ode_parsercdf.P	14,326	5/19/03 1:50 PM
37	::ode_props.P	784	4/7/02 6:54 AM
38	::ode_utils.P	13,891	3/28/03 10:38 AM
39	::ode_utilscdf.P	13,947	5/16/03 1:01 PM
40	::odeconstructor	522,007	
41	:::class_documentation.P	1,631	4/11/03 11:23 AM
42	:::clstree.P	17,755	8/4/03 10:49 AM
43	:::com	200,550	
44	:::xsb	200,550	
45	:::odeconstructor	186,480	
46	:::AttributeValueShow.form	4,055	3/21/03 7:44 AM
47	:::AttributeValueShow.java	4,143	3/21/03 7:44 AM
48	:::ConstructorPanel.form	5,347	5/13/03 9:56 AM
49	:::ConstructorPanel.java	17,872	5/13/03 9:56 AM
50	:::DBLoginValidator.java	2,533	3/28/03 7:09 AM
51	:::ExpandablePanel.form	2,277	8/7/02 10:00 AM
52	:::ExpandablePanel.java	6,802	8/7/02 10:00 AM
53	:::FileFilters.java	2,731	7/23/02 12:24 PM

1	::::::FindTextDialog.form	6,859	1/27/03 7:05 AM
2	::::::FindTextDialog.java	7,663	1/27/03 7:05 AM
3	::::::GTOptionChooser.form	6,598	3/14/02 9:37 AM
4	::::::GTOptionChooser.java	11,499	3/14/02 9:37 AM
5	::::::GTWrapper.java	889	7/2/02 5:42 AM
6	::::::ODEDesktop.java	24,330	5/13/03 10:03 AM
7	::::::OptionChooser.form	6,598	2/14/02 7:00 AM
8	::::::OptionChooser.java	10,598	2/14/02 7:00 AM
9	::::::PageViewer.java	22,135	5/1/03 12:45 PM
10	::::::Settings.java	1,707	4/1/03 11:57 AM
11	::::::TokenPane.form	2,862	2/27/02 4:37 AM
12	::::::TokenPane.java	8,186	2/27/02 4:37 AM
13	::::::dbview	30,796	
14	::::::DBPanel.form	520	5/2/03 5:13 AM
15	::::::DBPanel.java	5,476	5/2/03 5:13 AM
16	::::::DBView.form	8,570	5/13/03 2:01 PM
17	::::::DBView.java	10,951	5/13/03 2:01 PM
18	::::::QueryTableModel.java	5,279	5/17/02 4:46 AM
19	:::::oms	5,778	
20	::::::Concept.java	1,313	12/12/01 5:56 AM
21	::::::OMSConstants.java	1,107	4/22/02 12:38 PM
22	::::::OMSInterface.java	3,358	5/9/03 5:19 AM
23	:::::util	8,292	
24	::::::SortedComboBoxModel.java	2,262	12/12/01 5:59 AM
25	::::::SortedListModel.java	2,793	5/17/02 4:52 AM
26	::::::SortedTableModel.java	3,237	12/12/01 5:50 AM
27	:::condition_editor.P	19,655	6/6/03 6:37 AM
28	:::domain_inferer.P	8,174	6/23/02 11:35 AM
29	:::domaineditor.P	23,415	1/30/03 7:25 AM
30	:::explanations.P	5,959	4/7/03 9:09 AM
31	:::export_ont.P	6,865	5/9/03 7:47 AM
32	:::extractor_view.P	927	5/1/03 12:44 PM
33	:::guis.P	11,634	4/7/03 9:09 AM
34	:::images	25,243	
35	::::A-table.gif	358	3/17/03 12:34 PM
36	::::Q-table.gif	346	3/17/03 12:34 PM
37	::::about.html	365	6/5/03 11:47 AM
38	::::icon_odeconstructor_line.gif	934	11/6/02 5:56 AM
39	::::icon_odeconstructor_line.jpg	899	11/6/02 5:43 AM
40	::::icon_odeconstructor_line32.gif	1,167	11/5/02 7:29 AM
41	::::icon_odeconstructor_line32.jpg	1,480	10/31/02 11:14 AM
42	::::odec_splash.jpg	19,556	6/5/03 11:35 AM
43	::::text.gif	138	3/28/03 7:21 AM
44	:::ode_domain_types.P	9,704	6/16/03 5:01 AM
45	:::ode_extr_utils.P	9,201	5/1/03 1:10 PM
46	:::odec_utils.P	20,301	5/9/03 7:49 AM
47	:::odeconstructor_init.P	13,109	6/13/03 12:33 PM
48	:::odetypes	10,464	
49	::::data_omsext.P	386	4/22/02 12:55 PM
50	::::schema_omsext.P	10,078	4/15/03 10:33 AM
51	:::rel_rules_editor.P	31,644	4/8/03 11:08 AM
52	:::rellist.P	40,012	6/12/03 1:19 PM
53	:::rule_templates.P	21,164	4/1/03 6:31 AM

1	:::type_rules_editor.P	15,692	4/24/03 7:34 AM
2	:::typetree.P	12,480	4/28/03 7:13 AM
3	:::value_abbreviations.P	13,329	8/4/03 10:49 AM
4	:::xjode.P	3,099	6/5/03 9:29 AM
5	:::odescan.P	6,106	1/7/03 11:54 AM
6	:::suptok.P	20,385	7/31/03 8:10 AM
7	:oms_matcher	1,866,107	
8	:::bin	7,255	
9	:::build.xml	7,255	8/18/03 1:57 PM
10	:::changelog	28,992	8/14/03 6:06 AM
11	:::generic_matcher._bat	1,058	8/18/03 1:54 PM
12	:::generic_matcher.c	861	1/3/03 7:34 AM
13	:::images	347,813	
14	:::About16.gif	644	8/26/02 5:04 AM
15	:::AppIcon16.jpg	900	11/6/02 5:55 AM
16	:::AppIcon32.gif	1,249	11/5/02 7:30 AM
17	:::ContextualHelp16.gif	198	3/20/00 5:14 AM
18	:::Delete16.gif	208	8/26/02 5:04 AM
19	:::Empty16.gif	832	9/12/02 7:33 AM
20	:::GreenLed13.gif	142	8/6/00 11:22 PM
21	:::Import16.gif	311	8/26/02 5:04 AM
22	:::Information16.gif	661	9/19/02 12:55 PM
23	:::Open16.gif	228	8/26/02 5:04 AM
24	:::Preferences16.gif	207	8/26/02 5:04 AM
25	:::RedLed13.gif	142	8/6/00 11:22 PM
26	:::Refresh16.gif	244	9/19/02 12:55 PM
27	:::Remove16.gif	213	3/20/00 5:15 AM
28	:::Save16.gif	206	8/26/02 5:04 AM
29	:::Save24.gif	266	3/20/00 5:15 AM
30	:::SaveAll16.gif	252	8/26/02 5:04 AM
31	:::SaveAll24.gif	334	3/20/00 5:15 AM
32	:::SaveAs16.gif	255	8/26/02 5:04 AM
33	:::Status16.gif	423	3/20/00 5:15 AM
34	:::Zoom16.gif	303	3/20/00 5:15 AM
35	:::splash_window.bmp	300,054	4/18/03 1:13 PM
36	:::splash_window.gif	16,751	11/7/02 5:51 AM
37	:::splash_window.jpg	22,790	2/28/03 5:36 PM
38	:::looks.jar	317,029	8/8/03 7:54 AM
39	:::matcher_docs	821,648	
40	:::index.html	2,620	9/30/02 9:10 AM
41	:::man_pages	818,921	
42	:::CVS	689	
43	:::Entries	590	7/18/03 10:11 AM
44	:::Repository	54	9/20/02 7:23 AM
45	:::Root	45	5/20/03 6:23 AM
46	:::images	788,176	
47	:::CVS	639	
48	:::Entries	533	7/18/03 10:11 AM
49	:::Repository	61	9/20/02 7:23 AM
50	:::Root	45	5/20/03 6:23 AM
51	:::desc.jpg	59,907	11/5/02 11:48 AM
52	:::detail_results.jpg	214,120	11/5/02 11:53 AM
53	:::help.jpg	109,501	9/23/02 8:37 AM

1	:::::menus.jpg	18,901	9/19/02 12:55 PM
2	:::::newclass.jpg	17,766	9/23/02 7:42 AM
3	:::::newrel.jpg	13,783	9/23/02 7:57 AM
4	:::::prefs.jpg	39,660	11/5/02 11:46 AM
5	:::::results.jpg	60,636	11/5/02 11:49 AM
6	:::::right_panel.jpg	119,989	9/19/02 12:55 PM
7	:::::select_obj.jpg	93,861	9/30/02 9:19 AM
8	:::::trees.jpg	39,413	9/19/02 12:55 PM
9	:::::page1.html	742	9/26/02 6:42 AM
10	:::::page10.html	1,349	9/26/02 6:42 AM
11	:::::page11.html	4,731	11/5/02 12:59 PM
12	:::::page12.html	1,074	11/5/02 12:25 PM
13	:::::page13.html	3,144	9/30/02 8:29 AM
14	:::::page2.html	1,391	9/26/02 6:41 AM
15	:::::page3.html	1,932	9/26/02 6:41 AM
16	:::::page4.html	4,755	9/30/02 9:21 AM
17	:::::page5.html	1,106	9/26/02 6:41 AM
18	:::::page6.html	1,488	11/5/02 12:57 PM
19	:::::page7.html	3,743	11/5/02 12:27 PM
20	:::::page8.html	3,110	11/5/02 1:03 PM
21	:::::page9.html	1,491	9/26/02 6:41 AM
22	:::matcher_images	107	
23	:::::CVS	107	
24	:::::Entries	3	9/20/02 7:23 AM
25	:::::Repository	59	9/20/02 7:23 AM
26	:::::Root	45	5/20/03 6:23 AM
27	:::matcher_fx	101,080	
28	:::compare_measures.P	7,213	11/8/02 11:59 AM
29	:::equals.P	333	9/5/03 1:14 PM
30	:::lcs	1,706	
31	:::::CVS	143	
32	:::::Entries	52	7/18/03 10:11 AM
33	:::::Repository	46	9/20/02 7:23 AM
34	:::::Root	45	5/20/03 6:23 AM
35	:::common_tokens.P	1,563	9/19/02 12:57 PM
36	:::least_common_subseq.P	4,632	11/8/02 11:59 AM
37	:::match_function_interface.P	4,313	2/28/03 11:10 AM
38	:::match_object_interface.P	16,446	8/14/03 5:58 AM
39	:::matcher.P	14,388	9/5/03 1:15 PM
40	:::matcher.P.old	16,451	2/6/03 8:10 AM
41	:::matcher_progress.P	741	2/19/03 6:26 AM
42	:::notequals.P	489	1/3/03 1:26 PM
43	:::phone_fax.P	7,413	11/11/02 12:35 PM
44	:::removedup.P	146	1/2/03 5:10 AM
45	:::tmp	13,983	
46	:::matcher_lite.P	13,348	11/6/02 9:27 AM
47	:::notequals.P	489	11/11/02 6:56 AM
48	:::removedup.P	146	11/11/02 6:56 AM
49	:::trie_matcher.P	4,779	11/8/02 11:59 AM
50	:::xj_matcher_progress.P	1,107	4/25/03 2:06 PM
51	:::zip_code.P	6,940	11/8/02 11:59 AM
52	:::matcher_gui	216,003	
53	:::logger_config.P	578	5/8/03 10:30 AM



1	:::matcher_func.P.102302	10,319	11/15/02 9:09 AM
2	:::matcher_fxconfig.P	3,706	2/28/03 11:10 AM
3	:::matcher_import.P	4,460	11/11/02 12:35 PM
4	:::matcher_io.P	10,655	2/28/03 11:10 AM
5	:::matcher_main.P	39,687	9/5/03 1:16 PM
6	:::matcher_match.P	9,285	4/29/03 2:07 PM
7	:::matcher_nodes.P	7,125	4/22/03 12:59 PM
8	:::matcher_obj_panel.P	6,314	4/22/03 1:02 PM
9	:::matcher_panel_utils.P	2	3/14/03 6:58 AM
10	:::matcher_panels.P	22,244	8/14/03 6:02 AM
11	:::matcher_pprules.P	26,649	4/22/03 1:02 PM
12	:::matcher_prefs.P	16,954	2/28/03 11:10 AM
13	:::matcher_result_panels.P	16,375	9/5/03 1:14 PM
14	:::matcher_tests.P	2,263	1/21/03 11:55 AM
15	:::matcher_utils.P	19,016	8/14/03 6:03 AM
16	:::matcher_validation.P	20,371	4/24/03 7:41 AM
17	:::matcher_omsext	18,083	
18	:::data_omsext.P	898	3/21/03 12:51 PM
19	:::schema_omsext.P	8,166	4/25/03 12:41 PM
20	:::schema_omsint.P	9,019	1/28/03 5:03 AM
21	:::runMatcherMain.bat	816	8/14/03 6:25 AM
22	:::xsb_compiler.P	937	3/4/03 6:53 AM
23	:utils	64,747	
24	:::marginals.P	1,081	6/11/03 6:22 AM
25	:::morphology.P	3,608	8/5/03 1:48 PM
26	:::repl_code.P	2,488	6/21/00 2:49 PM
27	:::singularize_table.P	5,763	7/28/03 4:43 AM
28	:::stdscan.P	5,850	3/24/00 11:34 AM
29	:::stdspell.P	2,157	3/24/00 11:34 AM
30	:::stdsupertok.P	9,624	5/31/02 6:46 AM
31	:::stdutils.P	29,802	8/4/03 4:42 AM
32	:::updateOs.P	1,633	3/2/01 5:34 AM
33	:::wnutils.P	1,236	9/13/00 7:31 AM
34	:::xed.P	79	10/5/00 3:44 PM
35	:::xeddis.H	38	10/5/00 3:44 PM
36	:::xeddis.c	1,388	10/5/00 3:44 PM
37	:wordnet	28,864,348	
38	:::eurika.P	2,075	8/23/00 6:55 AM
39	:::extractorGUI.P	8,378	8/23/00 6:55 AM
40	:::id_trans.P	2,978,154	8/23/00
41	6:55 AM		
42	:::prog.P	4,917	8/23/00 6:55 AM
43	:::queries.P	11,815	8/23/00 6:55 AM
44	:::wn_ant.P	176,876	8/23/00 6:55 AM
45	:::wn_at.P	23,388	8/23/00 6:55 AM
46	:::wn_cs.P	4,032	8/23/00 6:55 AM
47	:::wn_ent.P	8,113	8/23/00 6:55 AM
48	:::wn_fr.P	318,722	8/23/00 6:55 AM
49	:::wn_g.P	8,486,856	8/23/00
50	6:55 AM		
51	:::wn_hyp.P	1,464,815	8/23/00
52	6:55 AM		
53	:::wn_mm.P	206,122	8/23/00 6:55 AM

1	::wn_mp.P	122,849	8/23/00	6:55 AM
2	::wn_ms.P	12,741	8/23/00	6:55 AM
3	::wn_per.P	164,249	8/23/00	6:55 AM
4	::wn_ppl.P	2,070	8/23/00	6:55 AM
5	::wn_preds.P	3,539	8/23/00	6:55 AM
6	::wn_s.P	5,345,081		8/23/00
7	6:55 AM			
8	::wn_sa.P	73,550	8/23/00	6:55 AM
9	::wn_sim.P	416,119	8/23/00	6:55 AM
10	::wn_su.P	5,545,110		8/23/00
11	6:55 AM			
12	::wn_su2.P	3,474,840		7/25/03
13	12:25 PM			
14	::wn_vgp.P	9,937	8/23/00	6:55 AM
15	:xjcdfwidgets	225,785		
16	::abbreviationPanel.P	8,110	8/29/03	9:37 AM
17	::abbreviationPanel_calls.P	29,663	8/29/03	10:26 AM
18	::addRelation.P	2,899	8/4/03	4:42 AM
19	::cdfDisplayUtils.P	1,235	7/17/03	2:21 PM
20	::cdf_validation	63,902		
21	:::samplerData.P	11,574	6/6/03	10:37 AM
22	:::samplerData_lq.P	13,192	7/3/03	6:57 AM
23	:::validationGuiTemplates.P	39,136	9/9/03	12:17 PM
24	::componentWidgets.P	8,891	9/2/03	11:49 AM
25	::delimiterchoice.P	869	7/24/03	11:45 AM
26	::desktop_utils.P	4,764	8/12/03	4:41 AM
27	::export_format_calls.P	12,799	7/16/03	11:43 AM
28	::export_mask_editor.P	2,974	7/31/03	10:23 AM
29	::export_objects.P	15,394	8/1/03	1:18 PM
30	::external_form.P	5,977	8/8/03	8:11 AM
31	::import_objects.P	17,634	7/28/03	5:17 AM
32	::inference_rules.P	7,667	8/12/03	1:47 PM
33	::rel_widget.P	13,948	7/15/03	6:54 AM
34	::showProperties.P	9,538	8/4/03	4:42 AM
35	::suptokParseTreeGui.P	5,418	7/25/03	10:03 AM
36	::toolbar_widget.P	9,596	8/8/03	9:18 AM
37	::tree_templates.P	4,507	8/4/03	4:42 AM
38				

39           Appendices A and B will be incorporated into a substitute  
40 specification by preliminary amendment.

1                   DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

2

3           Turning now to Figure 1, a web agent creator 10 according

4 to the invention includes a user interface 12 based on a web

5 browser, an XPath discovery algorithm 14, a results editor 16,

6 an agent generator 18, and a form value editor 20. The user

7 interface 12 is based on "Webclient", a java embedding interface

8 of the Mozilla web browser. The user interface 12 communicates

9 with the other components to provide the nodes of the document

10 object model (DOM) that correspond to the text selected in the

11 user interface. The XPath discovery algorithm 14 supplies an

12 XPath to the agent generator 18 via the results editor 16. The

13 XPath is used by the agent 22 to extract the highlighted

14 information as well as similar information from the page. Those

15 skilled in the art will appreciate that an XPath is one type

16 of a pattern expression and that the discovery algorithm 14 can

17 be adapted to work with any type of pattern expression.

18

19           Given a collection of XPath expressions, the results editor

20 16 extracts matching text from the page and displays this text

21 in a table. The user can then provide names for the columns of

22 data in the table, choose to ignore data columns, or supply

23 examples of undesirable data that were erroneously extracted.

1       The agent generator 18 is used to build the agent 22 based  
2   on where the user has navigated in the web browser user  
3   interface 12 and the settings the user has provided in the  
4   results editor 16. The agent generator 18 uses DOM events to  
5   determine what the user is doing. In addition, the agent  
6   generator 18 enforces rules that ensure an invalid agent is not  
7   created. For example, users are not allowed to follow a link on  
8   a page unless they have created an XPath to extract that link  
9   (which the agent will use to locate the link). The agent  
10   generator 18 also warns the user when a page is encountered that  
11   contains formatting that is not supported by the agent, e.g.  
12   frames or JavaScript.

13  
14       When a user submits a form on a web site, the form value  
15   editor 20 displays all the parameters of the form that will be  
16   submitted and allows the user to set options for the agent 22  
17   when it submits the form. Options include which form parameters  
18   to submit and whether the value should remain constant or is  
19   supplied dynamically while the agent is running (e.g. from a  
20   database).

21  
22       From the foregoing, those skilled in the art will  
23   appreciate that the heart of the web agent builder 10 is the  
24   XPath discovery algorithm. The algorithm uses examples provided

1 by the user to discover an XPath expression that will extract  
2 all of the highlighted data as well as similar data from the DOM  
3 tree representing the HTML page. For example if the user  
4 highlights a fragment of a table then the XPath expression  
5 learned from this fragment will extract all the elements in the  
6 table.

7  
8 Conceptually, the data items in a web page can be  
9 partitioned into different sets of homogeneous records. For  
10 example, a table row or a list element in the document can be  
11 considered as a record and all the rows in a table as a set of  
12 homogeneous records. To identify such records, certain types of  
13 DOM tree elements are designated as grouping nodes (e.g. `<tr>`  
14 and `<li>`). From the highlighted examples, the agent creator 10  
15 learns a set of data extractor XPath expressions to locate data  
16 items rooted under grouping nodes. However, such an expression  
17 can be very general and will match unrelated data items. For  
18 example, if the user highlights a fragment of a table from which  
19 he wishes to extract data and the document contains two tables  
20 then the data extractor expression can potentially match the  
21 data elements in the rows of both the tables. One can avoid  
22 matching unwanted data by confining the scope of applicability  
23 of the data extractor expression to subtrees that contain the  
24 desired data. To locate such subtrees certain elements in the

1 DOM tree are designated as locator nodes. The grouping nodes in  
2 the subtree rooted at a locator node contain related data items.  
3 E.g. a <table> element is a locator node since all the rows in a  
4 table occur as grouping nodes in its subtree. The XPath  
5 discovery algorithm learns an isolator XPath expression to  
6 isolate each grouping node occurring in the subtree rooted at  
7 the locator node. The data extractor expressions are then  
8 applied separately to each subtree rooted at a grouping node.  
9 Hence, these isolator expressions serve to group the nodes  
10 matched by the data extractor expressions (such as all the  
11 <tr>'s in a <table>). The set of nodes that were found by  
12 applying the data extractor expressions on a document from a  
13 given grouping node are considered to be part of the same group,  
14 or record (e.g. all the data items in a <tr>). Figure 2  
15 illustrates this concept.

16  
17 In Figure 2 the <tr>'s are grouping nodes and <table> is  
18 the locator node. It is also important to note that isolator  
19 expressions serve to make data extraction expressions resilient.  
20 This is because the context for applying them is confined to a  
21 particular region (defined by the locator node) and is hence  
22 immune to changes that occur in a web document that is outside  
23 this region.

1       As an illustration, assume that the user highlights the  
2 text region corresponding to the leaf nodes in Figure 2. Then  
3 the isolator XPath that will be generated by the XPath discovery  
4 algorithm will be `//table/tr` and the set of data extractor  
5 expressions will be `./td[1]/text()`, `./td[2]/text()`, and  
6 `./td[3]/text()`. The isolator expression will match any node  
7 that is labeled `<tr>` and is the immediate child of a node  
8 labeled `<table>`. Figure 2 illustrates what is matched by these  
9 expressions and how the results are logically grouped. It  
10 should be noted that both isolator and data extraction  
11 expressions are learned from the same set of examples.

12  
13       The learning algorithm takes as its input a set of nodes  
14 corresponding to the highlighted items in the web browser user  
15 interface, identifies the locator and grouping nodes based on  
16 these data items and learns the corresponding data extractor and  
17 isolator expressions. These expressions form the essence of a  
18 navigation map for the web agent (22 in Figure 1). When the web  
19 agent is launched these expressions are applied to a page as  
20 follows:

21       First the web agent will find all nodes that match the  
22 isolator expression. This will identify the grouping nodes  
23 under a locator node.

Second, for each such matched grouping node, the web agent will find a match for each of the data extractor expressions applied from the matched node. This will extract all of the related items within the subtree rooted at the locator node.

The navigation map built using the web agent creator (Figure 1) is an XML document consisting of one or more page maps as well as a series of actions that interconnect the page maps. Each page map includes isolator and extractor XPath expressions indicating which parts of a web page contain meaningful information to be extracted, along with basic formatting information for formatting output. Actions can be defined to follow links or to fill out forms as necessary to reach the page(s) containing the desired data.

When a web agent (22 in Figure 1) is launched, the agent interprets the map, performs the actions indicated and formats the extracted data as called for in the map. According to the presently preferred embodiment, the output formats include XML and MS Access Database format, either of which may be selected by the user during the creation of the web agent. Figure 3 is a screen shot illustrating the main user interface for the web agent creator.



Turning now to Figure 3, the web browser interface 12 of the web agent creator has two toolbars: a horizontal toolbar 30 and a vertical toolbar 32. The horizontal toolbar 30 is similar to a standard web browser having back and forward buttons, 30a, 30b respectively, a stop loading button 30c, and a refresh page button 30d. Button 30e is a browser security button which allows the user to specify information about security settings, proxy servers, etc. The vertical toolbar 32 includes web agent creation functions. The button 32a begins the process of creating and recording an agent. The button 32b launches and executes agents. The button 32c stops the agent creation process. The button 32d allows the user to add an example of the types of information to be extracted by the agent. The button 32e shows the results of data extraction in the form of a table.

Figures 4a-4d illustrate the steps involved in creating a web agent with a web agent creator according to the invention. Starting at Figure 4a, the user enters the URL of a web site containing data of interest, and the page corresponding to the URL is then loaded. Next, the user instructs the web agent creator to start recording the user's actions by clicking on the button 32a. The user follows a link in Figure 4a to the page containing data of interest, e.g. the page shown in Figure 4b.

1 The web agent creator records that this link must be taken and  
2 then loads the corresponding page.

3  
4 When the page containing data of interest is displayed, the  
5 user highlights a sample of the data for extraction as shown by  
6 the highlighting in Figure 4b. From this highlighted fragment,  
7 the XPath discovery algorithm learns a set of Xpath expressions  
8 and displays all of the data items that it matches in the page  
9 as shown in Figure 4c. At this point the user can add more  
10 examples by clicking button 32d, start over by clicking button  
11 32a, or save the agent by clicking the button 32c

12  
13 The Xpath expressions learned by the XPath discovery  
14 algorithm depend on the region of the page highlighted by the  
15 user. By adjusting the highlighted region the user can quickly  
16 generate an expression that will exactly match all of the items  
17 of interest. When the user is ready to save the agent by  
18 clicking on button 32c, recording is terminated, the navigation  
19 map is generated and the save dialog box is displayed as shown  
20 in Figure 4d. The navigation map encodes information about the  
21 links to be followed to navigate to this page and the Xpath  
22 expression that will be applied to the page for extracting the  
23 data of interest.

1       To extract data from a web site the agent built for that  
2 site must be launched. According to an alternate embodiment of  
3 the invention, a separate agent manager runs the agent by  
4 interpreting the agent's site map. The agent manager is a  
5 powerful Java based desktop tool for managing and executing a  
6 society of agents. The agent manager allows the user to add,  
7 delete, and schedule agents to run at user specified times and  
8 with a given regularity. This tool also enables the user to  
9 specify the input and output of the completed agent tasks and  
10 have them presented and stored in a structured and coherent  
11 fashion on the user's desktop. The agent manager has a task  
12 wizard that steps users through selecting an agent, determining  
13 data input and output locations, and specifying when and how  
14 often a task should be performed. The agent manager also has a  
15 preferences dialog box where users can set parameters for their  
16 agent operations including the agent directory, the task  
17 directory, security settings, and proxy settings.

18  
19       The tools of the invention also include a text extractor  
20 tool for extracting semi-structured data, such as tabular data  
21 from text documents residing locally or over the Internet.  
22 Those familiar with text files, particularly text files which  
23 were created by scanning a paper document, will appreciate that  
24 tabular data is not structured with easily recognizable

1 delimiters. For example, a structured table in a word  
2 processing document, e.g. MS Word, normally delimits columns  
3 with tabs and rows with carriage returns. Columns of tabular  
4 data in text documents are often separated by a non-constant  
5 number of spaces. Figure 5 is an example of financial data  
6 presented as a table in an unstructured text form.

7

8 Referring now to Figure 5, extracting text data from text  
9 tables that are irregular poses several difficulties. In broad  
10 terms, irregularity is characterized by variable length data  
11 items (perhaps spanning multiple words) that possibly overlap  
12 with items in neighboring columns. For example in Figure 5, the  
13 item "BERKSHIRE HATHAWAY INC DEL CL" in the 1st column (under  
14 header "NAME OF ISSUER") of the 7th row overlaps with the item  
15 "COMMON STOCK" in the 2nd column (under header "TITLE OF CLASS")  
16 of the 1st row. Such irregularities arise during machine  
17 conversion of data from one format into another. For example  
18 when pdf documents are converted into text form using xpdf, the  
19 text tables corresponding to their pdf counterparts appear  
20 drastically misaligned.

21

22 A simple approach for extracting items is to find fixed  
23 separators between successive columns. Intuitively, a fixed  
24 separator is a unique position (with spaces running through all

the rows at that position) that distinguishes items occurring in a pair of neighboring columns. As shown in Figure 5, it is not always possible to find fixed separators. Even if fixed separators exist, it is unclear how they can unambiguously separate columns that have multiword items (e.g. column 1 in Figure 5).

Another technique that is sometimes used for extracting data from text is based on regular expressions. Regular expressions specify patterns that occur in text and a regular expression processing engine extracts pieces of text that match the specified patterns. Although regular expression based extractors are powerful when dealing with text processing in general, they are quite cumbersome and difficult to use in the presence of tables consisting of items that span several words and/or overlap with items in other columns.

Although the columns in Figure 5 are not delimited by fixed separators, by visual inspection a casual observer can still correctly associate each item with its corresponding column. This is because all of the items belonging to a column, despite having irregular alignments, appear clustered more "closely" to each other than to items appearing in different columns. Although such clusters can be clearly discerned by a human

1 observer, making them machine recognizable is the key to robust  
2 automated extraction of data items from text-based tables.

3  
4       Clustering enables the present invention to make  
5 associations between items in a column based not merely on  
6 examining items in adjacent rows but across all of the rows in  
7 the table. This means that even though an item in a row may not  
8 appear to be in the correct column when examined in isolation  
9 (due to misalignments), when viewed in the context of all the  
10 rows it can be associated with the correct column (e.g. "CL" in  
11 the 7th row appears to be in the 2nd column although its correct  
12 association is the 1st column). Although clustering techniques  
13 abound in the literature for various application domains such as  
14 data mining, information retrieval, and image processing, its  
15 use in text table extraction has not been previously explored.

16  
17       Given the rows of a text table as the input, the text  
18 extractor of the present invention associates items in the table  
19 with their corresponding columns using a clustering based  
20 algorithm. Each line is broken down into a set of tokens, each  
21 of these tokens being a contiguous sequence of non-space  
22 characters. Based on the positions of these characters in the  
23 line, a center for every token is computed. The center of any  
24 token in a cluster is assumed to always be closer to the center

1 of some other token in the same cluster than it is to a token in  
2 an adjacent cluster. Inter-cluster gaps are the spaces between  
3 the end tokens in adjacent clusters. Starting with an initial  
4 partitioning of the set of tokens into clusters, the partition  
5 gets refined in every iteration. Refinement amounts to creating  
6 larger clusters by merging adjacent clusters based on inter-  
7 cluster gaps. The text extraction algorithm terminates when no  
8 further refinement is possible.

9  
10 A text extractor of the invention is implemented in Java  
11 and consists of approximately about 3000 lines of code. The  
12 clustering algorithm assumes that each column is associated with  
13 a unique header, each header being a string consisting of one or  
14 more words, abbreviations or numbers. Typically, the text table  
15 can be logically separated into two consecutive regions, namely  
16 the header region consisting of all the headers followed by the  
17 data region. The header region may span multiple lines and the  
18 two regions are separated by special tokens, which may include  
19 spaces. The user supplies a list of keywords appearing in the  
20 headers as well as the separator tokens. Prior to invoking the  
21 clustering algorithm, the text extractor performs header  
22 discovery to identify the headers of the columns in the table.

1       The clustering algorithm of the invention can also be used  
2 to find tables in a text document using a closeness metric  
3 between rows and the spatial location of gaps in the lines.  
4 Based on this metric and the observation that rows in a table  
5 are clustered together, the iterative clustering algorithm  
6 described above can be used to detect tables as well as extract  
7 structured data from them.

8  
9       In product and financial data domains tables typically are  
10 generated from templates. The user-defined parameters, based on  
11 sample text tables, are used by the text extractor of the  
12 invention to extract tabular data automatically from a batch of  
13 files containing similar tables.

14  
15       Figures 6a and 6b show part of the user interface of the  
16 text extractor of the invention. It can be seen that there are  
17 four tabbed panels, two of which (extractor panel and separator  
18 properties) are shown in Figures 6a and 6b. The other panels  
19 allow for selection of input and output files. Input files are  
20 text files in the form shown in Figure 5, i.e. badly formatted.

21  
22       After the input and output files have been selected and  
23 named, the extractor panel shown in Figure 6a is used to provide  
24 a list of all the header names (column keywords) that could



possibly occur across all the tables in the files. The header names can be either single words or phrases. Note that there is an option for removing bad (long) lines that span all of the columns, as well as an option for turning on error detection which then automatically separates all of the input files into two directories, one for correctly extracted tables and the other for all those where extraction failed.

In the separator panel shown in Figure 6b, the user also provides a list of tokens that serve as separators between the column headers and the data. It also has input fields for tokens which serve as delimiters for the table (both at the top and at the bottom). This allows the text extractor tool to filter out text which is not part of the table.

The text extractor can automatically perform the extraction without any manual intervention. However, to increase the yield of correctly extracted tables, it is sometimes desirable to supply the minimum column gap as a parameter. In this case, the system is used interactively by the user to sample a few text tables to estimate this gap. The clustering algorithm will not merge adjacent clusters if the gap between them is larger than this parameter value.

1       After running the system and performing extraction over a  
2 collection of text tables, the user can examine the directory  
3 containing incorrectly extracted tables, sample a few of them,  
4 identify and determine whether the incorrectly extraction was  
5 caused by an erroneous estimate of minimum column gap, re-adjust  
6 the configuration parameter, and begin a new run on all these  
7 tables.

8  
9       Figure 7 illustrates an output (in MS Excel format)  
10 generated by the text extractor for the table illustrated in  
11 Figure 5. The present embodiment of the text extractor can  
12 generate output in several formats including bar-separated, MS  
13 Excel and MS Access.

14  
15       As mentioned above, the tools of the invention also include  
16 an ontology management system (OMS), an ontology directed  
17 classifier (ODC), and an ontology directed extractor (ODE). The  
18 OMS manages a highly structured knowledge base which indicates  
19 hierarchies of classes where classes have properties that are  
20 given as relationships to other classes (ontologies). Here the  
21 terms properties, attributes and relationships are used  
22 interchangeably. The ODC uses the hierarchy of classes, e.g. a  
23 taxonomy such as the UNSPSC (United Nations Standard Products  
24 and Services Code) or the NAICS (North American Industry

Classification System). The ODE uses the properties of the hierarchy of classes to extract attribute-value pairs from free flowing text.

Referring now in more detail to the OMS, it is currently implemented in XSB Prolog and provides an efficient way of storing and managing classes and their relationships, as well as objects and their attributes. The OMS provides a structure to represent knowledge. The knowledge is represented as a set of objects, a particular collection of sets of those objects, relations among these sets and objects, and constraints on those relations.

For example, the UNSPSC taxonomy is a classification system to guide global commerce. It is a 8 digit hierarchical schema consisting of 4 levels in the hierarchy, plus an additional 2-digits (9/10 position) that are optional. It enables users to consistently classify the products and services they buy and sell.

The UNSPSC has approximately 50 main classes. One of these classes is "42: Medical Equipment and Accessories and Supplies" A fragment of this class and some subclasses is illustrated in Figure 8. This class of medical equipment is subdivided into 20

subclasses, one of which is "4231: Wound Care Products". The Wound Care Product class has 10 subclasses, one of which is "423117: Suture and related products", which in turn has 8 subclasses, one of which is "42311701: Sutures". In a slightly simplified example, this hierarchy is represented in the OMS using a class relation, containing:

```
Class(4231, 'Wound Care Products').
```

And a direct\_subclass relation, containing:

```
Direct_subclass(42311701,423117).
```

Representation of complex real-world problems requires more than just taxonomic information. For example it is not enough just to know that an item is a suture; it is important to know details about the suture, such as whether it is absorbable or not, what kind of needle it has, its length, etc. To represent this kind of information the OMS supports the specification of attributes (or relationships) associated with classes. For example, the 42311701: Suture class may have these attributes: Material, Surface Treatment, Needle Style, Needle Type Designator, etc. According to the invention, each attribute must take its value from an appropriate domain. For example, the Material attribute of sutures may have 'GUT' and 'SILK' as its domain of values. The values that are appropriate for a particular attribute are themselves a class, and appear in the

OMS as a class. Note that the value domains shown in Figure 8 are not necessarily complete and only illustrate the first one or more values in the value domain. Attributes are generally pre-defined by an imported ontology, but can be defined by the user through a manual process. An OMS relationship connects the attributes of a particular class with the value domains appropriate for the attributes. For example:

```
relationship('Sutures','Material','Suture Materials').
```

In addition to containing information on classes, an OMS contains information about objects, that are members of the classes. An OMS according to this example may have information about a particular suture, say procurement item 12345, described by "SUTURE ABS SURG SZ 3-0 4.50" LG GUT UNARMED MED TREATED STER 12S". This is an object and information about it is stored in the OMS in an object relation, `object(12345,'procured object')`. The class in the OMS to which it belongs is represented in a `direct_memberof` relation, e.g., `direct_memberof(12345, 42311701)`, indicating that this object is a member of the class of Sutures. Attributes of particular objects (consistent with the attribute types declared in the relationship facts) are represented in the OMS using the attribute relation, as in: `attribute(12345,'Material','Gut')`.

Rules are given to define a "memberof" relation, which is transitive through classes, which would imply, for example, that object 12345 is a member of the class of Wound Care Products. Rules are also provided to support inheritance of attributes (when the attribute is declared to be inheritable.)

The OMS supports primitive classes, including character strings, integers, and floating point numbers. It also supports parameterized classes, represented by terms, and representing semantically the cross product of a set of more basic classes. Inheritance is also supported through parameterized classes. In addition, the OMS supports a primitive data type of "clause", which allows rules to be saved in the OMS.

The OMS has strong similarities to other systems for managing object-oriented knowledge such as Flora, RDF and Protege. However the OMS differs from these systems in that it can easily convert between Prolog and database syntax. In addition, the OMS does not have a constraint that a given attribute is unique or functional. Finally, the OMS does not provide for non-monotonic inheritance of relationship facts for default reasoning. These differences make the OMS more efficient and easier to use in commercial systems.

1       A taxonomy determines an organization for a set of objects.  
2   For example, given a particular partially known product, the  
3   first step in relating it to the known products is to determine  
4   what class in the UNSPSC taxonomy it should be placed in; or  
5   more precisely, it should determine all classes in the UNSPSC  
6   taxonomy of which it is a member. Given a taxonomy and a short  
7   description of an object believed to belong to that taxonomy,  
8   the ODC finds the nodes in the taxonomy containing that object.  
9   Even if the exact smallest class of which it is a member cannot  
10   be determined, the ODC may be able to determine some set of  
11   classes of which it is likely to be a member. Since every class  
12   in the taxonomy is a subset of the class above it, such a set of  
13   classes will form a sub-tree of the taxonomy. A sub-tree is  
14   referred to as a "cone". The problem that the ODC solves, is to  
15   take an arbitrary taxonomy and an arbitrary description of an  
16   item covered by the taxonomy and produce a "cone" that is the  
17   best guess of where that item should be classified in the  
18   taxonomy. The following examples illustrate how the ODC works.

19

20       Example 1:

21       A large company buys products from a great many suppliers.  
22   This company has a policy of trying to make at least 15% of all  
23   their purchases from minority suppliers. So whenever a company  
24   buyer is trying to find a supplier for a new need, the buyer is

1 encouraged to try to find a minority supplier. However, finding  
2 a new supplier can be difficult.

3  
4 To help in this process the ODC can classify all known  
5 minority suppliers to a standard taxonomy according to a  
6 description of supplier capabilities. There is a standardized  
7 taxonomy for classifying suppliers of parts and services, the  
8 NAICS taxonomy (North American Industry Classification System),  
9 which can help organize these minority suppliers. There are  
10 databases of minority suppliers in which they have self-  
11 classified themselves to NAICS categories. With this database,  
12 a buyer can go to the category of suppliers that supplies the  
13 product or service needed and contact suppliers classified  
14 there. However, there are over 2500 categories in the NAICS  
15 taxonomy, so finding the correct one(s) is a daunting task. By  
16 using the ODC, a buyer can enter a brief description of their  
17 need, and the classifier can return a relevant cone of the NAICS  
18 taxonomy, thereby greatly focusing the buyer's search.

19  
20 Example 2:

21 It is known that self-classification of the supplier to  
22 NAICS categories is a highly error prone practice. The ODC can  
23 be used to minimize or eliminate these errors. Suppliers can  
24 use the ODC when classifying their company's products and



1 services. They can enter a description of their products and  
2 services and then choose the best matching description from the  
3 cone(s) that the ODC returns. In this way, the purchasers can  
4 obtain far more accurate data regarding suppliers.

5  
6 Example 3:

7 Another application of the ODC is what is known as "spend  
8 analysis". Given a taxonomy of products (such as UNSPSC), the  
9 ODC can be used to classify all products that a large company  
10 buys. Then products classified to the same category can be  
11 analyzed and compared to determine where most of the cost is  
12 incurred, and whether the best deals are always obtained for  
13 these similar products.

14  
15 The ODC operates in several stages. The first is referred  
16 to as Taxonomy Token Weighting. Taxonomy descriptions are  
17 tokenized and super-tokens are created by applying replacements  
18 that are predetermined or have been entered by the user. This  
19 step may include elimination of certain tokens that are  
20 irrelevant for classification, such as the tokens "the", "a",  
21 etc. These super-tokens can be used to correct obvious  
22 misspellings, to account for common abbreviations, and so on.  
23 For each super-token T, the non-normalized weight of T is taken  
24 to be total occurrences of all super-tokens in the taxonomy

divided by the total occurrences of T in the taxonomy. This weighting gives higher weight to tokens that occur less frequently in the taxonomy, and thus are likely to be more useful for classification.

The second stage is referred to as Node Weighting for Descriptions. The object description is super-tokenized, and a weight is derived for each node in the taxonomy as a function of the super-tokens in the description that match the nodes description, their position in the descriptions, and the co-occurrence of multiple tokens.

The third stage is referred to as Weight Propagation and Normalization. Given the semantics of taxonomies as being ordered by set-inclusion, the classification weight of a node N is taken to be the weight of N as determined in the second stage, together with the sum of the weights of all of its children. In this stage previous weights are propagated and then normalized so that the weight of the root of the taxonomy equals 1.

The fourth stage is referred to as Determining the "Best" Node and Cone. Based on the results of stage three, a search starts at the root and descends the tree to determine the "best"

1 match for the object description based on the node's normalized  
2 match weight obtained in stage three. Users of the ODC may use  
3 various parameters to determine when the descent should stop  
4 along with various relaxations of the aggressiveness of the  
5 descent.

6

7 It can be shown that the normalized weights produced by  
8 stage three form a probability measure when the taxonomy graph  
9 is a tree.

10

11 ODC can be tuned in two basic ways. First, the  
12 classification algorithm is heavily dependant on the weights of  
13 super-tokens as determined in stage one. By tuning the super-  
14 tokenizer, which is applied both to taxonomy nodes and to object  
15 descriptions, the various weights of nodes can be affected.  
16 Second, training items can be provided. These are descriptions  
17 that are pre-classified to their correct taxonomy node.

18

19 Training items are treated as if they extended the  
20 taxonomy, being taxonomy nodes that are immediate children of  
21 the node in which they are properly classified. Then the  
22 processing proceeds on this "larger" taxonomy. The only  
23 difference in treatment is that in stage four, when the best  
24 node is determined, these new training items are excluded from

1 being chosen. So training descriptions "pull" similar  
2 descriptions toward themselves.

3  
4 As may be appreciated from the description of the ODC  
5 above, the ODC has several components. The lowest-level  
6 component of ODC is called the super-tokenizer. It reads the  
7 input descriptions (of a taxonomy node or of an item  
8 description) and applies user-specified rules to construct the  
9 "words" used for matching items with nodes. The super-tokenizer  
10 handles simple user-specified abbreviations, but also supports  
11 more complex standardization of units of measure, or number  
12 range mappings. It includes a fully recursive rewriting system.  
13 The output of the super-tokenizer is a set of "words" and their  
14 associated weights. At this point in the process, the weights  
15 depend only on the location of the word in the input string.  
16 This allows the classifier (under user control) to increase the  
17 weight of words at the beginning (or the end) of the  
18 description. The inventors have found that a more accurate  
19 classification of short descriptions is often possible by  
20 weighting early-appearing words higher than later-appearing  
21 words.

22

23 The second component of the ODC utilizes the results of the  
24 super-tokenizer to build weight tables for all words appearing

1 in the taxonomy node descriptions. Words are weighted based on  
2 the frequency of their occurrence and their location in the  
3 taxonomy. When it is given a description string, the ODC uses  
4 the weight tables to determine a weight for each node in the  
5 taxonomy, the weight indicating a confidence of that taxonomy  
6 node being the correct node for the description. The weights  
7 are standardized and accumulated, and then used to determine an  
8 optimal cone for the description. The width and depth of the  
9 cone are controllable by parameters set by the user.

10  
11 The final component of the ODC is a graphical user  
12 interface (GUI) that allows a user to load a taxonomy, load  
13 descriptions to classify, optionally load pre-classified  
14 training descriptions, tune the classifier with abbreviations  
15 and more training items, classify descriptions and view the  
16 resulting cones and weights. It also supports manual correction  
17 of misclassified items and the exporting of classified batches  
18 of data.

19  
20 The classification subsystem is coded in XSB (an extension  
21 of Prolog) code, and the user interface is implemented in a  
22 proprietary interface generation system called XJ, which is  
23 implemented in Java. XSB and XJ communicate using an open-  
24 source subsystem, called InterProlog.

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23

The super-tokenizer has been designed and built to efficiently support a very large number of complex replacements using trie-indexing. In addition, the compact code of the XSB language allows larger data sets to be stored in memory. The open source semantic dictionary, WordNet, has been used to generate some replacements to allow the classifier to incorporate semantic information.

Figure 9 shows a screen shot of the ODC in a typical use. The lower right panel shows a list of descriptions automatically classified to nodes in the NAICS taxonomy. Highlighted is a description "food service, bottled sauce sales, catering, hospitality" which was automatically classified to the "caterers" node. The upper right panel shows an explanation of the classification, displaying the computed cone, a sub-tree of the NAICS taxonomy. The numbers in the square brackets at the beginning of the node labels provide the "confidence" that the node is correct for the selected goods and services. The words in square brackets at the end of the node labels provide the words that match and their relative contribution to the confidence.

1       The ODE is used to extract attribute-value pairs from  
2 unstructured textual descriptions. For example, suppliers  
3 typically store product descriptions in a single "description"  
4 field of a table or a database or a simple text list such as:  
5 "SUTURE ABS SURG SZ 3-0 4.50'' LG GUT UNARMED MED TREATED STER  
6 12S". Such descriptions make aggregation and parametric  
7 searches for equivalent items very difficult due to the fact  
8 that different suppliers use different features and vocabularies  
9 to describe similar items. Ontology Directed Extraction (ODE)  
10 enables automated extraction of standardized features and values  
11 such as:

12  
13       ITEM\_TYPE="SUTURE, ABSORBABLE, SURGICAL", END\_TYPE="UNARMED",  
14 MATERIAL="GUT", SURFACE\_TREATMENT="CHROMMSALT", SIZE="3-  
15 0", LENGTH="4.50 INCHES", STERILITY="STERILE" AND PACKAGE\_INFO="12S"  
16

17       A great deal of information is available in the form of  
18 natural language descriptions which are very difficult to reason  
19 about using automated tools. For example, it is very difficult  
20 to program an automated tool to determine whether <Silk Suture,  
21 2-0, 18", FS> and <Nonabsorbable 18 inch 2.0 Suture> are  
22 equivalent. Although both of them are classified as sutures,  
23 that does not necessarily mean that their properties are the  
24 same. The extraction process can be viewed as an OMS

1 transformation in which the extractor looks through attribute-  
2 value pairs of objects containing their natural language  
3 description and then populates other attribute-value pairs for  
4 the objects. Having extracted those attribute-value pairs, it  
5 is then possible to export them to a database table and run  
6 queries on them, have another application that makes use of  
7 them, or store them in the OMS. For example, from the  
8 description above the extractor determines: Material - Silk,  
9 Size Designator - 2-0, length - 18 inches, Needle Type  
10 Designator - FS, and even infers that the Type is Nonabsorbable  
11 (as all Silk sutures are nonabsorbable).

12

13       It is important to know what class the object belongs to  
14 before running it through the extractor, as different classes in  
15 ontology have different attributes, and the extractor needs to  
16 know what attributes are valid for the description. For  
17 example, from the description RED LEAD the extractor would  
18 extract Lead Polarity (Positive) if the object were classified  
19 in Electrical Devices. However, if the object is in a paint  
20 class, the extractor would determine paint color (Red) and  
21 chemical base (Lead).

22

23       The goal of the ODE is to make it possible for domain  
24 experts, not computer technologists, to create extractor



1 programs that extract attribute-value pairs from unstructured  
2 descriptive text. For this, it is necessary to specify  
3 "parameters" to a generic extractor, and provide easy ways for  
4 the domain expert to specify these parameters.

5  
6 The parameters include an ontology (e.g. Figure 8) which  
7 contains information about classes of objects and their  
8 properties and types of values. The ontology also contains  
9 information about abbreviations and special word usages, their  
10 contexts of use, and preferences. The other extractor parameter  
11 is a set of pattern rules, that provide information about how  
12 non-enumerated values for attributes can appear in text input.  
13 These declarations provide information about how attribute  
14 values can appear in text input.

15  
16 The present ODE is built on the XSB tabled logic  
17 programming engine, whose powerful grammar and unification  
18 capabilities make it an excellent platform for this kind of  
19 processing.

20  
21 To create an ODE extractor, a domain expert must create (or  
22 refine) the ontology (adding new types, if necessary.) The  
23 present invention provides an example-based easy-to-use tool  
24 that a domain expert can use for constructing extractors.

1  
2       From the foregoing it will be appreciated that the ODE has  
3 two main parts, the ODE Constructor and the ODE Launcher (or ODE  
4 Extractor Builder as referenced in Figure 13). The ODE  
5 Constructor allows the user to easily train a generic extractor  
6 by populating ontology knowledge about classes and the ODE  
7 Launcher applies that knowledge to the batches of pre-classified  
8 objects to extract their attribute-value information from  
9 descriptions. Both of these parts are built on top of the ODE  
10 engine, making use of a scanner, an efficient super-tokenizer  
11 and simple grammar. Figures 10 and 11 illustrate the ODE  
12 Constructor GUI and Figure 12 illustrates the ODE Launcher.

13  
14       Operation of the ODE is illustrated by the following  
15 example. To perform an extraction, the user must first build  
16 the extractor knowledge base with the ODE Constructor by  
17 supplying relations for the class and their domains, as well  
18 abbreviations and other fine tuning information, all of which is  
19 stored in the OMS.

20  
21       Next, the user runs the set of descriptions (that have been  
22 previously classified using the Classifier) through the ODE  
23 Launcher (using the previously built OMS) and the launcher  
24 extracts values of attributes from those descriptions. The

1 process is iterative, if the user is not satisfied with the  
2 results he may continue adding extractor knowledge until he is  
3 satisfied with the results. The process is also repeatable, as  
4 the knowledge built can be used later with descriptions from  
5 different sources providing that they are classified to classes  
6 for which the extractor is built. Note that this may require  
7 some additional abbreviations and values if the data is  
8 substantially different. However, there is a finite number of  
9 ways that data can be presented in a text and generally, an  
10 extractor trained to work on descriptions from several diverse  
11 sources will extract most of the information from the  
12 descriptions of some new data source as well.

13  
14 The ODE Constructor allows the user to create an extractor  
15 for classes in an ontology, add relationships representing the  
16 properties that the given class might have, and add domain  
17 information for the relationships. It also allows the user to  
18 add abbreviations and replacement rules with a simple GUI  
19 interface. The ODE also provides ontology editing capabilities  
20 for a domain expert to use. To that end, the ODE Constructor  
21 has a training area section that allows the user to load Text,  
22 HTML, and MS Access or Excel tables and add values to the  
23 ontology by selecting text in the training area and pressing the  
24 Value button (+) in the "Domain Type" window or as Add as

1 abbreviation (+) in the abbreviation window. In addition, the  
2 user has the capability of checking whether the information has  
3 been correctly added by selecting some sample text and prompting  
4 the constructor to perform extraction on that text. This shows  
5 the user which attribute-value pairs can be extracted from the  
6 selected description. The user can also request to see what  
7 parts of the selected text do not contain any values according  
8 to the extractor's current base of knowledge. This helps the  
9 user test the extractor without running the whole description  
10 batch through the ODE Launcher.

11  
12 Value domains for attribute-value relationships are divided  
13 into two types - Enumerated and Parameterized. Enumerated  
14 domains represent a finite set of acceptable values. For  
15 example, the domain of Strand Fiber Arrangement for a Suture is  
16 an enumerated domain with values Braided, Monofilament,  
17 Multifilament and Twisted. However, the domain of suture length  
18 is a measure (which is a number with an associated unit such as  
19 2.5 meters) and it is difficult or impossible to name all the  
20 possible values it can take.

21  
22 By default, when a user creates a new relationship for a  
23 class, a new Enumerated Domain is created for this relationship.  
24 For example, if the user selects a Suture class and adds

1 Material as a relationship, a new Suture-Material-Domain is then  
2 created. The user can select this domain and add its possible  
3 values like Silk or Plastic by selecting those values in a  
4 sample text in the training area or by manually typing in the  
5 values. The values of the domain are presented in an ontology  
6 as subclasses of the domain class.

7  
8 The ODE Constructor also allows users to create and edit  
9 Parameterized domains. Parameterized domains are fairly complex  
10 and as such, some of the most frequently used parameterized  
11 domains are provided for the user with the creation of the OMS;  
12 they are integer, number and measure. The user can select a  
13 relationship and choose an option of changing its domain type to  
14 an appropriate domain from the OMS tree. However, sometimes the  
15 user will be required to create a new complex, previously non-  
16 existent value domain. For example, Size Designator for Sutures  
17 is generally represented as: 2-0, 3-0 and so on. A user can  
18 create a new parameterized type that has two arguments (for 2  
19 and 0) and add a replacement rule saying that if the extractor  
20 sees a pattern where an integer is followed by '-' (dash)  
21 followed by an integer, the ODE Constructor should take the  
22 first integer as the first argument and the second integer as  
23 the second argument. The process of creating a new domain is  
24 carried out in a convenient rule editor where the user does not

1 have to manually type in the rule, rather, the user constructs  
2 the rule by selecting an integer concept in the list of  
3 available classes, adding -(dash), and picking the integer class  
4 again.

5

6       The user can have several rules of the same type to  
7 recognize the various patterns that may appear; for example, the  
8 user can add a rule to extract 2.0 as the Size Designator 2-0.  
9 That would be a similar rule with integer followed by  
10 '.' (period) followed by integer. Such pattern recognizing rules  
11 allow the user to process a string, extract only the information  
12 that is desired and ignore the rest. For example, in a  
13 dimension 5 mm X 4.5 mm, the first measure is length and the  
14 second is width for some domains. So in creating a rule for  
15 length, the user will pick measure, select that he is interested  
16 in all of its arguments (which are 'Sign', Number and 'Unit'),  
17 add 'X' and pick the second measure and select that he is not  
18 interested in arguments of the second measure. This would  
19 create an internal replacement, which would result in the ODE  
20 seeing a string like '5 mm X 4.5 mm' and picking the first  
21 measure which is '5 mm' to be the value of the length property.  
22 Conditions may be specified in the rule editor to enforce that  
23 the first measure be length only if greater than the second  
24 measure.

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24

Rules can be edited at any time should the user wish to change them or add new patterns to existing rules. In general, there is a limit to the number of ways a value can be present in a string, therefore, only a few rules or patterns need to be constructed to recognize values of a certain domain in a text.

The ODE Constructor also provides for abbreviations and simple word-to-word replacements. Sometimes strings in the text can be found in an abbreviated or modified form. To recognize these abbreviated strings the user needs to add abbreviation information to the OMS. To add an abbreviation for a value, the user can either select corresponding text in the training area or can manually type in the abbreviation. In addition, the user can add an abbreviation for a word that is a substring of a value, for example, abbreviation DBL for word DOUBLE which is a substring of the value Double-Armed. This is carried out within the framework of a convenient abbreviation-table editor. Abbreviations are contextualized, i.e. they can be applied for a description classified to a certain node and all its subclasses but not other nodes. For example, STR might mean Sterile in a medical domain and strength (of material) in a vehicle domain. By default, the context of an added abbreviation is the class for which extractor knowledge is being built.

1  
2       The user can edit and select a higher or lower class in the  
3 ontology to be a context of an Abbreviation. Certain  
4 replacements that are not abbreviations but rather are inference  
5 rules may be added along with abbreviations. For example, if  
6 Silk is in the text for Suture the ODE can infer that the Suture  
7 Type is Non-absorbable. Some abbreviations that are used often  
8 and that apply to all classes are provided for the user when he  
9 creates a new OMS for the extractor.

10  
11       In addition, the ODE Constructor provides for the concept  
12 of Dialect where the abbreviation occurs. Dialect represents  
13 the source of data specific for an abbreviation. For example,  
14 all descriptions coming from one web site might have S1 as an  
15 abbreviation for Absorbable suture, however that abbreviation  
16 may not apply to other sources. By default, an abbreviation  
17 will apply to all sources, but specific dialects can be  
18 specified by the user.

19  
20       A Preferences feature allows the user to fine tune ODE to  
21 resolve any preferences. For example, if PLASTIC SURGERY is in  
22 the text we want to infer that Needle Design Designation is  
23 Plastic Surgery, not that the Material is Plastic. There is an  
24 easy preferences mechanism presented in the table of preferences



for an Extractor Class (Suture in the example), where the user can add a preference to prefer Needle Design Designation over Material, providing that the string from which the former is extracted is longer then the string from which the latter is extracted.

Note that it is not necessarily desirable to set preferences for values of all domains. For example, if the ODE finds GAUZE BANDAGE it might be desirable to get both bandage type as Gauze Bandage and material Gauze (the Bandage type domain is not a repetition of the material domain though it includes the word 'gauze' here, other values for the type are Surgical Rubber Bandage, Elastic Bandage, Adhesive bandage). So in this case it is not necessary to add a preference and the extractor, by default, will extract all the possible values from the string (both the material and the type in this case).

There are two types of preferences that can be added to prefer one value over another when the first is extracted from a longer substring, or to have that preference when they are extracted from the same-length substring. For example, from the same string AC-25 both ID number and Needle Type Designator are extracted, however there can be a preference to prefer Needle Type Designator over ID number.

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23

When the domain expert decides that the extractor is sufficiently trained, he/she can process a batch of descriptions with the extractor using the ODE Launcher. The descriptions may be initially processed through ODC to determine their class. The user can then run the extractor and extract attribute-value pairs for each of the descriptions using the ontology. The results of the extraction are stored in an OMS that can be saved for further manipulations and queries, or they can be exported to an MS Access table or to a delimited text file.

If the user gathers information from a Web site and knows its classification (for example, going to a web site that displays only Suture information), he can load the data directly into the ODE Launcher for the class with appropriate relationships.

After the extraction is complete, the user may review the results of the extraction, and if the results are not satisfactory, can continue refining the ontology by using the ODE Constructor in an attempt to obtain more precise extraction results.

1       As shown in Figure 10, the main ODE constructor display  
2 window is divided into two sections - the upper panel, and the  
3 lower panel. The upper panel displays existing knowledge about  
4 classes and their properties in the form of an ontology, while  
5 the lower panel is used to load examples and training data into  
6 ODE. Within the upper panel there are four separate windows;  
7 the first of these (upper left) is the class window in which the  
8 user can see what OMS or Ontology they are working with and  
9 where they will be able to view a tree showing all of the  
10 classes needed for extractor construction. The second window is  
11 the Relationships window. This window displays information  
12 regarding the Relationships or Attributes of the selected class  
13 in the ontology the user is working with. The third window is  
14 the Values window. This window displays all of the information  
15 or values correlating to the Attribute selected in the  
16 Relationships window. Finally, the forth window is the  
17 Abbreviations window, which displays all of the possible  
18 abbreviations that the ODE will recognize for the selected  
19 Values in the third window.

20  
21       The lower panel training area is where the user loads  
22 sample files (Text, HTML, MS Excel or MS Access tables)  
23 containing example descriptions (e.g. the Bandages domain in  
24 Figure 10). It is not necessary to have such a file loaded.

1 However it is useful for reference and may serve to increase the  
2 ease and speed of the extractor creation process. To load a Web  
3 page, text, Access or Excel table in the training area the user  
4 presses the Load File button on the bottom-right corner of the  
5 screen.

6  
7 At any point during the process of building an ontology,  
8 the user can select a description in the training area text and  
9 press the Extract Attributes button (which is the first of two  
10 buttons with a table illustration located on the vertical tool  
11 bar running along the left side in the training-text area). At  
12 that point, extraction is performed on the highlighted  
13 description and the results in the form of the extracted  
14 attribute-value pair table is displayed for the user's review as  
15 illustrated in Figure 11. This aids the user by allowing  
16 him/her to see whether all of the values were extracted  
17 correctly and what values or domains need to be added or refined  
18 without requiring the user to process the entire batch of items  
19 through the ODE launcher. The user also has the capability of  
20 viewing what parts of the text did not provide any useful  
21 information for the extractor, and can then evaluate whether  
22 they contain interesting values. To do this, the user selects  
23 the show non-extracted substrings button (the second button on

the tool bar) which populates a table displaying the pieces of text that were not used in the extraction process.

If the user wants to create an ontology anew, he can start by loading an existing taxonomy that contains the Bandages class or can create a new class (Bandages) by clicking on the root of the new taxonomy then selecting the +... button under the class tree and entering the name of the new class. Then the user can enter relationships and corresponding domains of possible values. Looking at the sample text in the training area may help the user to understand the values of which properties are mentioned for the domain. For example, for Bandages the user sees that material and dimension are in the text and would want to extract those. For the material attribute, the user creates a Material relationship (property) by clicking the +... button in the section under the properties header and typing in the new Relationship name (Material). A new relationship Material is thus created with the corresponding domain of values Domain-Bandages-Material. However that domain is empty and the user must supply possible values for that domain. In this example, those values would be Plastic, Rubber, Flexible Foam, etc. To add the values, the user presses the +... button in the domain section and types in a value for the Domain-Bandages-Material. Alternatively, the user can select such a value in the training

1 text and press the +... button (located in the domain section  
2 third window) that will add a selected text as a value of the  
3 domain. After some values have been added, the user can add  
4 abbreviations for those values. That is done by selecting a  
5 value (for example, Rubber) and pressing +... button in the  
6 abbreviation section and typing in the abbreviation (for  
7 example, Rbbr).

8

9 The user can check whether the added values are extracted  
10 correctly. For example selecting 'Active Strips Flexible Foam  
11 Bandages, 1"X3"' text and pressing the Extract Attributes button  
12 displays the Material - Flexible Foam entry in the extracted  
13 attributes table. By adding relationships and values in this  
14 manner, the user can build the extractor trained for Enumerated  
15 domains.

16

17 The user can load domain values from a text file and avoid  
18 some of the burden of building domains. This is done by  
19 clicking on the load file button and selecting the file and the  
20 proper file delimiter. Thus, the user can load predefined  
21 values if they are available.

22

23 For values such as Width and Length to be extracted from  
24 descriptions like 1"X3", parameterized domains need to be

1 created or selected from existing ones (like measure in this  
2 case). The user can create and select the appropriate  
3 relationship (for example Length) in the relationships panel,  
4 right click on it and choose the 'Change domain to measure'  
5 option. Then a pattern rule can be added specifying that if  
6 measure is followed by measure then the second one (or first  
7 one) is length. In a similar way, the user can also create a  
8 new parameterized domain if there is no appropriate one in the  
9 ontology. Additionally, the user can also specify a condition  
10 in the pattern rule by showing that in the pattern the first  
11 measure is less than, or equal to, the second measure.

12

13 The ODE constructor also allows the user to save the  
14 ontology at any point and load it again for further processing  
15 at a later time.

16

17 After an extractor is built with the ODE Constructor, it is  
18 ready to be deployed for extracting attribute-value information  
19 from a batch of objects containing text descriptions.  
20 Extraction in the ODE Constructor is only done on a small set of  
21 descriptions to help populate the extractor knowledge base;  
22 extraction on large amounts of data is performed using the ODE  
23 Launcher. This simplifies the process of extracting attribute-  
24 value information from a batch of descriptions using the

1 extractor knowledge-base populated by the ODE Constructor. A  
2 screen shot of the ODE Launcher is shown in Figure 12.

3  
4 Once the user loads the ontology into the ODE Launcher, an  
5 item can be classified to a node in this ontology by the  
6 classifier. Using attribute information about this class from  
7 the ontology the extractor extracts all the attributes  
8 appropriate for that class from the item description. It is  
9 also possible to import objects with their descriptions from  
10 text files if the user already knows that the descriptions in  
11 those files are related to some extractor class, i.e. it has  
12 been pre-classified. For example, if the user obtained  
13 descriptions from a web site describing bandages, he can load  
14 them directly as objects of the Bandages class and can extract  
15 attribute values from bandage descriptions.

16  
17 Figure 12 shows extraction from descriptions of a batch of  
18 Bandage objects. The results are displayed in table form and  
19 can be exported to an Access table or a delimited text file or  
20 kept in OMS for further processing.

21  
22 Although the invention is designed as a process that will  
23 perform end-to-end extraction in an automated fashion, it is  
24 desirable to be able to assess and measure the quality of the



1 extraction performed. Based on the quality assessment, the user  
2 can fine-tune the system parameters appropriately. The  
3 invention includes a validation methodology to statistically  
4 sample extracted data and compare it to original data from the  
5 source in order to assess the quality of the extraction. In one  
6 embodiment the statistical sampling techniques were based on a  
7 well known quality measurement standard which defines an  
8 Acceptable Quality Level (AQL). According to a more recent  
9 embodiment the techniques are based on ANSI/ASQC Z1.4-1993 which  
10 defines an Acceptable Quality Level (AQL). AQL is defined as  
11 the number of defects per 100 items produced. For example, in  
12 the context of attribute extraction from unstructured product  
13 data descriptions, AQL indicates the number of erroneously  
14 extracted attributes per 100 records, each consisting of one  
15 product description. The procedure defined in ANSI/ASQC Z1.4 is  
16 to choose a random sample from a production run based on the  
17 size of the run. This sample is inspected for defects. To  
18 achieve a certain AQL, the sample can have at most a certain  
19 number of defects.

20  
21 The methodology for assigning AQLs according to the  
22 invention proceeds as follows. A random sample for a given  
23 extraction run is selected based on ANSI/ASQC Z1.4 tables. An  
24 initial AQL is selected and the sample is inspected manually.

1 If an acceptable level of defects are found, another random  
2 sample is obtained and a lower AQL is selected. This process  
3 continues until a sample with too many defects for the chosen  
4 AQL is found. The last successful AQL is taken to be the  
5 correct AQL for the extraction run. If, on the other hand, the  
6 initial sample fails, another random sample is obtained and a  
7 higher AQL is selected. The process continues in this way until  
8 a sample succeeds and the chosen AQL for the successful sample  
9 is taken to be the correct AQL for the extraction run.

11 Figure 13 is a simplified flow chart giving an overview of  
12 the operations of each of the parts of the invention. Turning  
13 now to Figure 13, the user utilizes the agent builder to provide  
14 examples of target information to be extracted from a web site.  
15 The agent builder generates an agent map, which is used by the  
16 agent manager to harvest the desired data from the specified web  
17 site. Upon harvesting, this information is evaluated by the  
18 agent validator to assure that the map has correctly, and  
19 completely, specified the location of the target data. If not,  
20 the map is edited and refined by the agent builder. If AQL is  
21 achieved, the harvested data is then classified in the ODC. The  
22 ODC was initially trained for a specific domain of interest in  
23 the Classifier Builder. The Classifier launcher assigns the  
24 best class to each of the object (item) descriptions. The

1 process us then validated, and if the desired AQL is achieved,  
2 the data is then passed to the ODE component. If the desired  
3 AQL is not achieved, the ODC is re-trained to improve upon  
4 classification results. In ODE, attribute-value pairs  
5 corresponding to a class of an object with a description are  
6 being extracted. For each ontology, information, along with  
7 abbreviations, and replacement rules are being supplied by the  
8 extractor builder. The Extractor launcher invokes the ODE logic  
9 engine to use ontology specific information to extract  
10 attribute-value pairs from object descriptions. Results are  
11 then validated and the extractor builder is used again should  
12 additional fine tuning and editing of the extractors be  
13 required. When data is structured in an ontology, a form  
14 matcher might be used to reason about the data and to determine  
15 product equivalence (similarity of content). The process may  
16 have to go through several iterations if it is determined  
17 (through the validation process) that the existing data is  
18 incomplete and information needs to be extracted from the WWW.

19  
20       There have been described and illustrated herein methods  
21 and software tools for acquiring data from diverse sources and  
22 organizing the data in a form that may be used by a database.  
23 While particular embodiments of the invention have been  
24 described, it is not intended that the invention be limited

1 thereto, as it is intended that the invention be as broad in  
2 scope as the art will allow and that the specification be read  
3 likewise. For example, the taxonomies and ontologies discussed  
4 herein are simply used as an example. The invention can be  
5 applied to any taxonomy and ontology. It will therefore be  
6 appreciated by those skilled in the art that yet other  
7 modifications could be made to the provided invention without  
8 deviating from its spirit and scope as so claimed.